Heuristic Evaluation of Labscape Version 0.2

Jeffrey Towle, Information School at the University of Washington
Sunny Consolvo, Intel Research Seattle
Evaluation date: May 8, 2002

IRS-TR-02-014

August, 2002

Heuristic Evaluation of Labscape Version 0.2

Jeffrey Towle, Information School at the University of Washington
Sunny Consolvo, Intel Research Seattle
Evaluation date: May 8, 2002

Evaluation Overview

Abstract
Labscape is a research project that investigates how computers can be tightly integrated with laboratory work. In Labscape version 0.2, a cell biology laboratory was instrumented with shared touch tablet computers that were distributed throughout the laboratory environment. Graphical representations of the biologist's plan were displayed on the tablets, which provided a way to access, capture, and organize data produced during laboratory work.

A heuristic evaluation was performed on Labscape version 0.2 to study the usability of the interface. A team of five evaluators examined each dialog of the interface, comparing each against a list of usability heuristics. If any aspect of the interface did not meet one or more of the heuristics, it was noted for this report. Fifty-five usability issues were identified by this evaluation method. Each usability problem has been catalogued, and includes a list of the heuristic(s) violated, a priority rating, a detailed explanation of the problem and, in many cases, an accompanying screen shot.

The evaluators
Eithon Cadag - recorder
Sunny Consolvo
Jeong Kim
Jing Su
Jeffrey Towle – moderator

Goal
Previous evaluations of Labscape (e.g., the lag sequential analysis performed in spring 2002) have focused on how well Labscape fit into the physical environment of the biology lab, not on the user interface (UI) itself. The goal of this evaluation was to uncover usability issues with the UI of Labscape version 0.2 to prepare for the upcoming summer 2002 UI redesign.

Intended Audience
The intended audiences of this report are the Labscape design/development team and other researchers following the progress of the Labscape project.

Table of Contents

Study Details

The heuristic evaluation (for a description of a heuristic evaluation, see Appendix A) of Labscape version 0.2 took place on May 8, 2002 in a conference room at Intel Research Seattle (see Fig. 1). Five evaluators were involved. The evaluation was performed in a group setting, with each evaluator contributing comments; one evaluator moderated and another recorded the discussion.

The evaluators were Eithon Cadag, Sunny Consolvo, Jeong Kim, Jing Su, and Jeffrey Towle. All evaluators were familiar with the target user groups for Labscape; none of the evaluators were members of the user groups. Only Jing Su, a member of the Labscape development team, had used Labscape prior to this evaluation. In addition to aiding with the evaluation, Jing acquainted the other evaluators with the UI before the evaluation began. The developer who designed the Labscape UI was not present at the evaluation, nor were any members of the target user groups.

This evaluation took two hours. During the first hour, the evaluators became familiar with the UI. The second hour was devoted to the evaluation itself. Jeffrey Towle moderated the session, Eithon Cadag served as the official recorder, and Jing Su operated Labscape.

The evaluation was performed using a large projected display. It should be noted that actual users of Labscape would not use such a display; this technology was for evaluative purposes only.

Figure 1: The conference room where the evaluation took place

Results

The result of this evaluation is the following list of usability problems. Each problem is accompanied by a priority rating, the name and number of the heuristic(s) violated (e.g., *system status(1)*), and a detailed description. Priorities are rated on a 5-point scale from 0 being "not an issue" to 4 being a "usability catastrophe" (see Appendix D for more details). For a complete list of the heuristics used, see Appendix B.

Startup and Login

1. ***Debug window* obscures *user check-in window***
   Priority: *cosmetic (1)*
   Heuristic violated: *system status (1)*
   Problem description: When Labscape first opens, the *debug window* appears directly on top of the *user check-in window*. The user must move the *debug window* to start using Labscape. This violates the *system status* heuristic because the user may not know that the system is waiting for him/her to login.

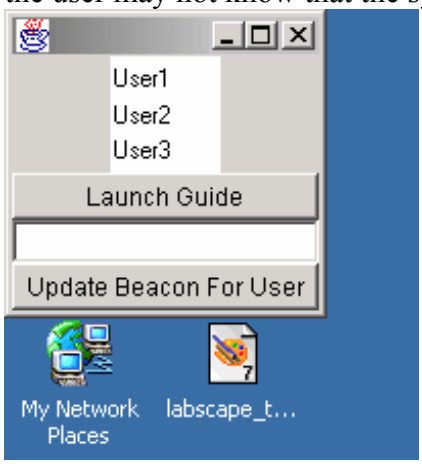

Figure 2: The *debug window* covers the *user check-in window*

2. **Default size *user check-in window* shows only three letters of the title**
   Priority: *cosmetic (1)*
   Heuristics violated: *consistency (4), recognition v. recall (6)*
   Problem description: The title is unreadable until the user stretches the window. This forces the user to either remember the function of the window or stretch it out to read the title.

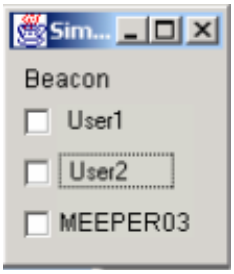

Figure 3: Default size of *user check-in window*

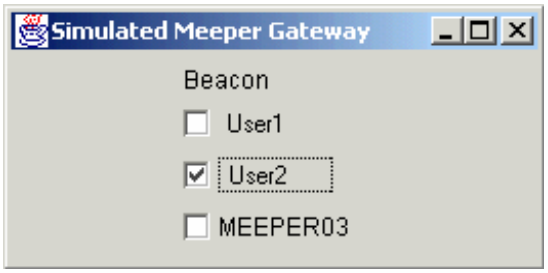

Figure 4: *User check-in window* stretched to accommodate the title

3. ***User check-in window* is titled "Simulated Meeper Gateway"**
   Priority: *cosmetic (1)*
   Heuristics violated: *real-world match (2), recognition v. recall (6)*
   Problem description: The title "Simulated Meeper Gateway" does not effectively communicate the window's purpose to the user, so the user must remember the window's function in order to log in. This title may also prove to be a source of confusion (e.g., if a user does not understand it, s/he may think s/he is using the wrong window).

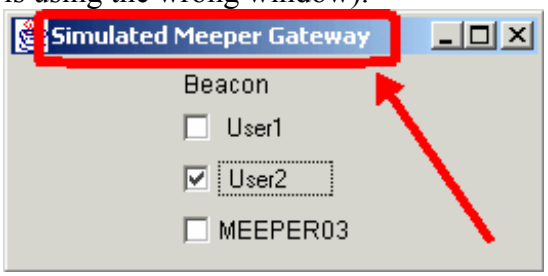

Figure 5: "Simulated Meeper Gateway" window title

4. ***User check-in window* checkboxes do not allow logout**
   Priority: *major (3)*
   Heuristics violated: *real-world match (2), consistency (4)*
   Problem description: The *user check-in window* uses checkboxes, which are traditionally used when multiple options can be selected simultaneously. With checkboxes, users expect to check an option to activate it and uncheck it to deactivate it. However, in Labscape, checkboxes do not function according to user expectations. Unchecking a box does not deactivate the option (i.e., log the user out). When a user wishes to log out, the user must manually close the window that s/he was using.

5. ***User check-in window* checkbox automatically logs in**
   Priority: *minor (2)*
   Heuristics violated: *system status (1), real-world match (2), consistency (4)*
   Problem description: As soon as the user has selected a name from the *user check-in window*, the *main window* of Labscape begins to load. In most programs, the user will expect to confirm a selection, particularly when checkboxes are used. The user will not expect Labscape to start automatically after only checking a box. The automatic login also makes it difficult for the user

to undo an accidental action because Labscape will load before any corrective action can be taken. Due to the lack of security, a user could easily log in as someone else by mistake.

6. **"Instructions" in the *user check-in window***
   Priority: *minor (2)*
   Heuristic violated: *real-world match(2), consistency (4)*
   Problem description: The word "beacon" does not indicate to the user what action should be taken in this window.

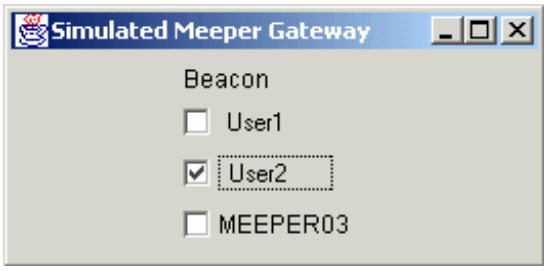

Figure 6: *Beacon* does not describe what action should be taken

7. ***User check-in* and *debug windows* block desktop icons**
   Priority: *cosmetic (1)*
   Heuristic violated: *minimalist design (8)*
   Problem description: These two windows open in the upper-left corner of the desktop, obscuring important system icons. NOTE: Because Labscape runs on dedicated tablets in the laboratory, this violation only applies to instances where the user is running Labscape at his/her desk.

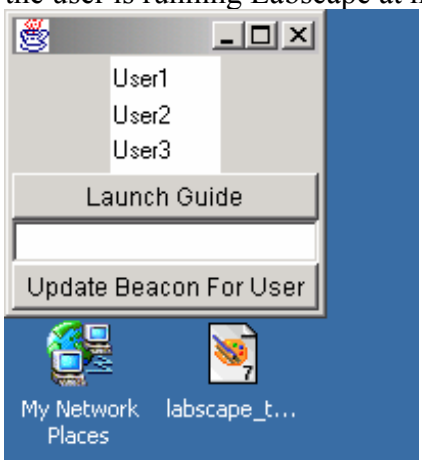

Figure 7: The *debug window* covers system icons

8. **No indication that the program is starting after a user logs in**
   Priority: *minor (2)*
   Heuristic violated: *system status (1)*
   Problem description: Once a user name has been selected from the *user check-in window*, Labscape gives no indication that the *main window* is loading. This problem is compounded by the lengthy loading time of the *main window*.

Main Window

9. **Initial screen on *main window* is blank**
   Priority: *major (3)*
   Heuristics violated: *system status (1), real-world match (2), consistency (4), error prevention (5), minimalist design (8)*
   Problem description: Once the *main window* loads, the user is presented with a mostly empty black screen. There is no indication that Labscape has finished loading or that it is waiting for user input. To get started, the user must know to click on the black screen to bring up the pie menu.

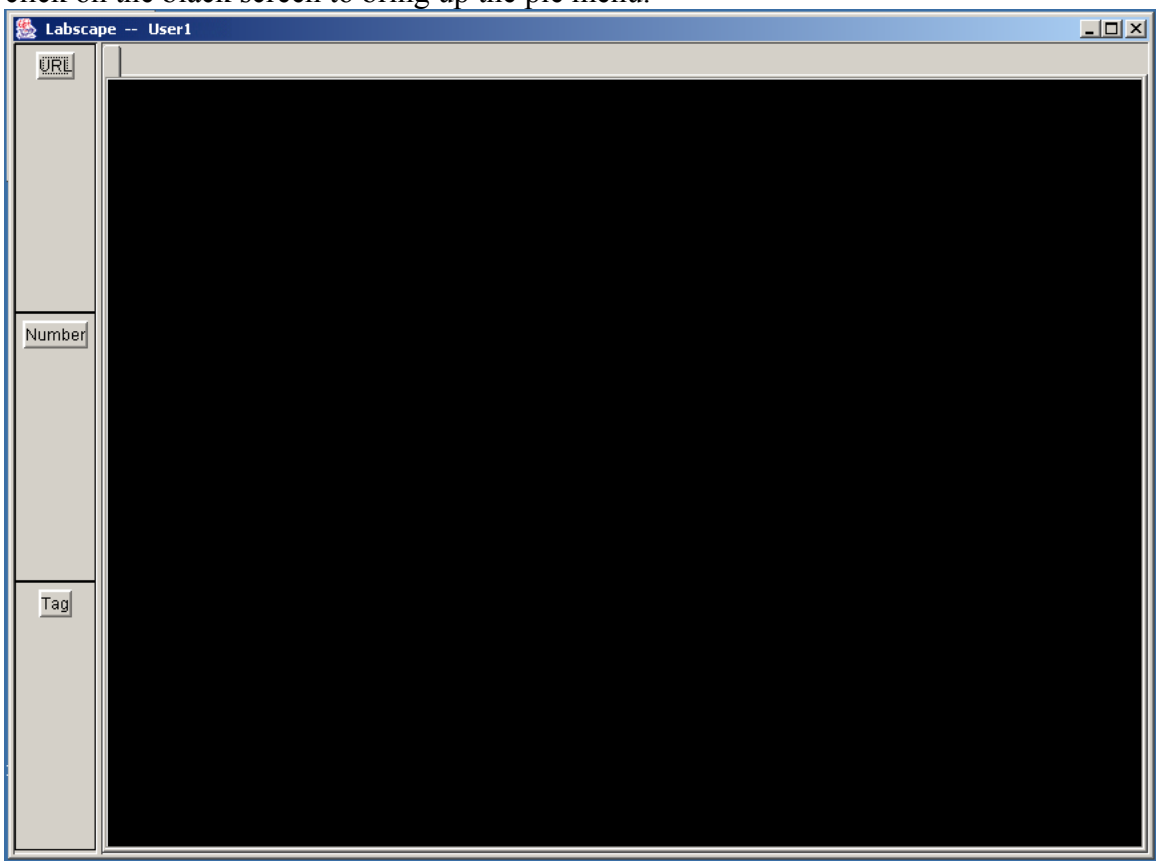

Figure 8: The initial *main window* is blank

10. **The *main window* background is black**
    Priority: *minor (2)*
    Heuristics violated: *minimalist design (8), consistency (4)*
    Problem description: The black background violates the principle of keeping a consistent interface. Because the target user groups are primarily Windows or Mac users, they may expect to see a white background on their computer applications. The three standard Windows buttons in the upper right corner suggest that Labscape will behave like any other Windows application, but the black screen suggests otherwise.

11. **Small blank navigation tab at the top of the screen**
    Priority: *cosmetic (1)*
    Heuristics violated: *consistency (4), minimalist design (8)*
    Problem description: When the main window opens, a small blank navigation tab is visible, though no experiment is open. This tab may confuse the user because, though visible, it does not contain any information or respond to mouse input.

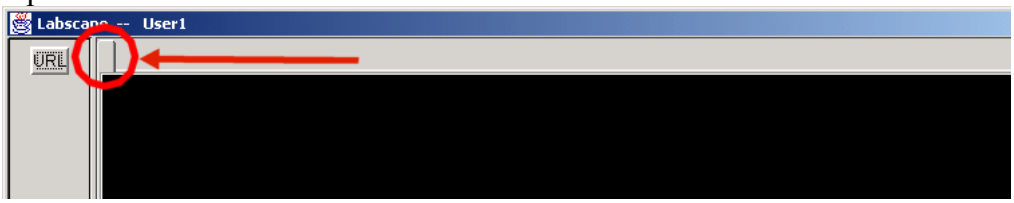

Figure 9: Empty navigation tab

12. **The tab representing the active session is not obvious**
    Priority: *minor (2)*
    Heuristics violated: *system status (1), consistency (4), minimalist design (8)*
    Problem description: Only a small white line differentiates the tab of the active window pane from those that are inactive, giving little indication as to which session is active.

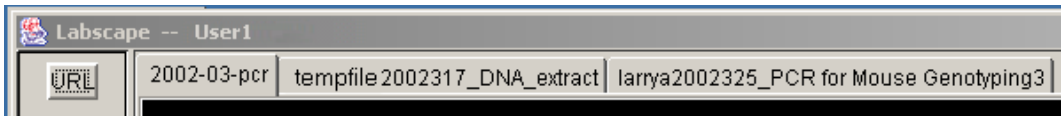

Figure 10: The active tab (2002-03-pcr) may be difficult to see

13. **New sessions do not appear in the foreground**
    Priority: *catastrophe (4)*
    Heuristic violated: *consistency (4)*
    Problem description: When the user opens a new experiment session, it loads in the background while the previous session stays in the foreground. If the user does not notice the new tab, s/he may not realize that the new session has loaded. S/he also runs the risk of mistaking the old session in the foreground for the newly-opened session.

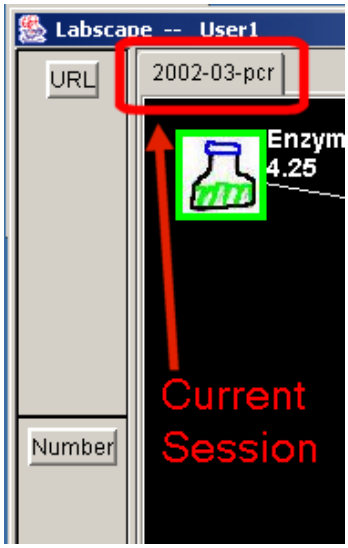

Figure 11: Tab configuration before opening a new session

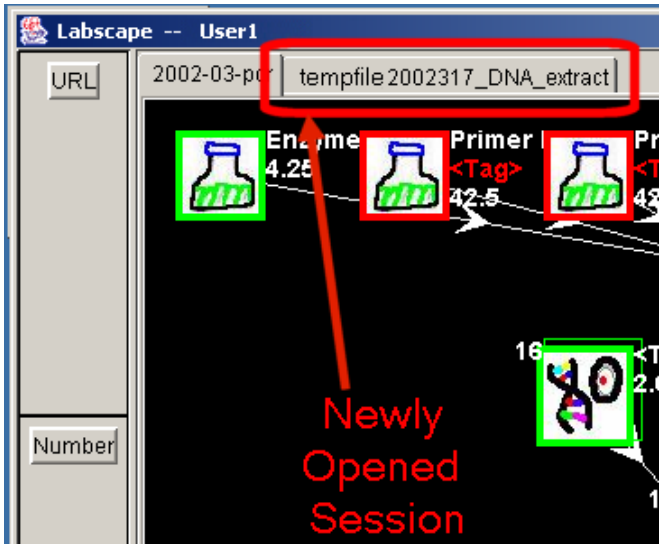

Figure 12: Tab configuration after opening a new session

14. **Navigation tab-length inconsistency**
Priority: *cosmetic (1)*
Heuristics violated: *consistency (4)*
Problem description: Tab length is dependent on session name length, resulting in different tab lengths.

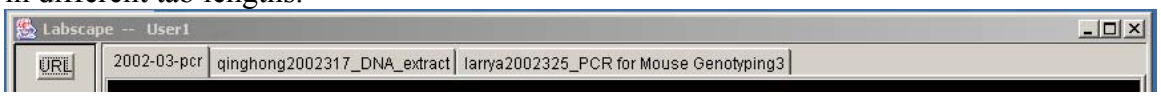

Figure 13: Each navigation tab is a different size

15. **Purpose of the left sidebar**
Priority: *major (3)*
Heuristics violated: *real-world match (2), user control (3), consistency (4)*
Problem description: The left sidebar may be confusing to the user because its function and purpose are not clear. The meanings of the labels (*Number*, *Tag*, etc.) are also unclear.

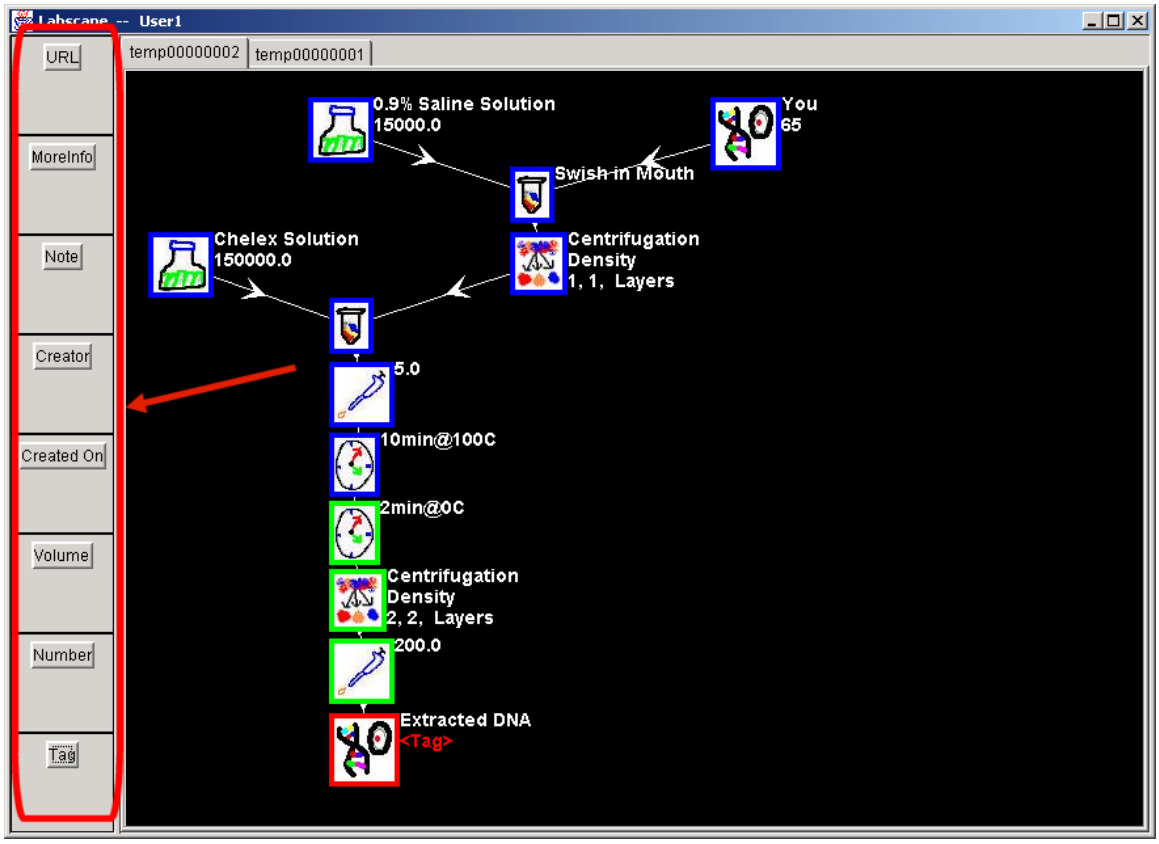

Figure 14: The left sidebar is a debug feature

Sample Flow Graph

16. **Overlapping of text, icons, and arrows**
Priority: *major (3)*
Heuristics violated: *system status (1), consistency (4), error prevention (5), minimalist design (8)*
Problem description: In the sample flow graph (SFG), icons, text, and arrows can overlap and obscure the view of other objects. The white text often overlaps icons or arrows (which are also white). Items become difficult to see because they are either completely obscured or the contrast between the objects is too low to differentiate. The icons can also overlap, potentially making it difficult for the user to differentiate between the text of one icon and the text of another. This may also be a problem because the screen shots are often used as experimental records (users print out the screen shots and paste them into their lab notebooks).

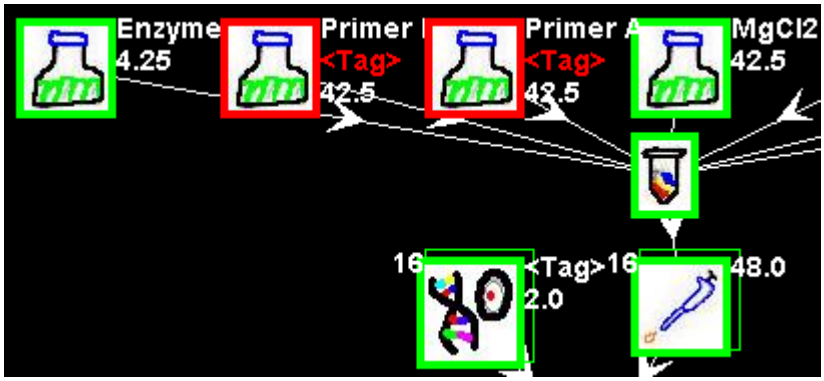

Figure 15: Icons, text and arrows often overlap

17. **Icon representations**
Priority: *major (3)*
Heuristics violated: *recognition v. recall (6)*
Problem description: There is no way to tell what each icon represents; the user must already know the meaning of the icon.

18. **Icon names do not match their dialog window titles**
Priority: *major (3)*
Heuristics violated: *consistency (4), real-world match (2)*
Problem description: The titles of some dialog windows do not match their icon names. For example, the *reagent* icon's dialog window is titled *Get Stock*. This discrepancy may make it difficult for the user to associate icons with dialog boxes.

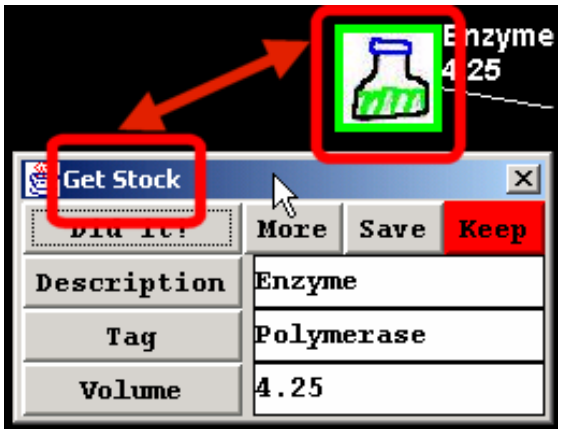

Figure 16: *Reagent* icon's dialog window is titled *Get Stock*

Icon	Icon Name	Dialog Title
	*Biological sample*	*Get Sample/Store Sample*
	*Combine*	*Combine*
	*Dispense*	*Split*
	*Detect*	*Store Sample/Measure/URL Measure*
	*Reagent*	*Get Stock*
	*Separate*	*Separate*
	*Incubate*	*Incubate*
	*Generic batch* (not used in practice)	*Empty Batch*

Figure 17: The icon name vs. the title of its dialog window

19. **Lack of association between dialog window titles and individual icons**
    Priority: *minor (2)*
    Heuristics violated: *system status (1), recognition v. recall (6)*
    Problem description: When multiple dialog windows are open, it becomes difficult to tell which pop-up window is tied to which icon, placing the burden on the user to remember these associations. If a user were to open the wrong icon, the lack of association between dialog windows and icons could prevent the user from realizing his or her error.

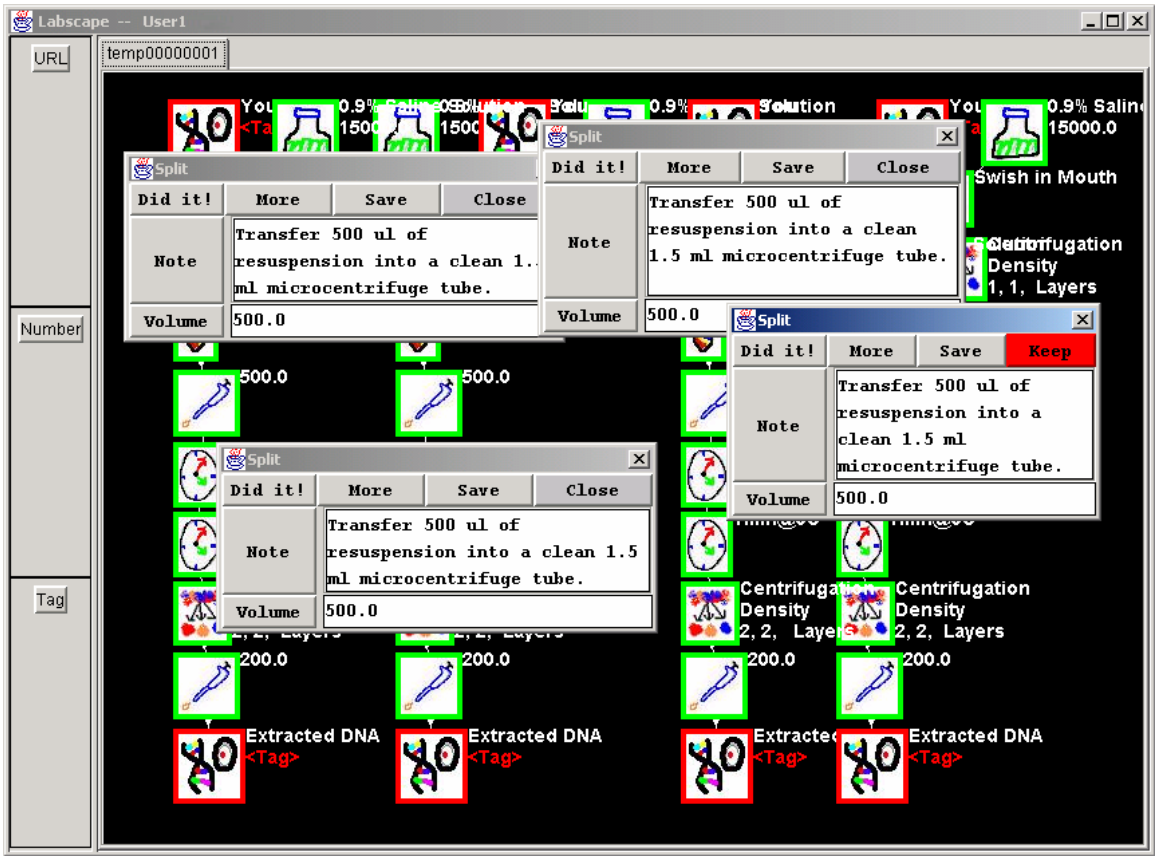

Figure 18: Matching individual icons to their dialog windows may be difficult

20. **Icons are different sizes and shapes**
Priority: *minor (2)*
Heuristic violated: *aesthetic design (8)*
Problem description: The icons come in different shapes and sizes.

Figure 19: Each icon is a different shape and size

21. **Use of red vs. green to differentiate state**
Priority: *major (3)*
Heuristics violated: *error prevention (5), error recovery (9)*
Problem description: Different states in the various steps of the SFG are only represented by the color of the icon's outline—red, green, or blue. (Red means that Labscape needs more information, green means that the step can be completed without any additional input, and blue indicates that the step has been completed). Users with red/green color deficient vision may not be able to distinguish between red and green states.

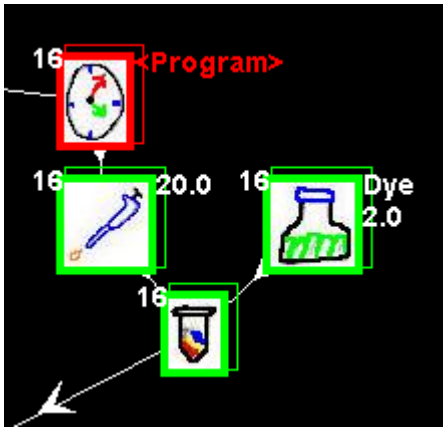

Figure 20: Red and green are used to show differences in status

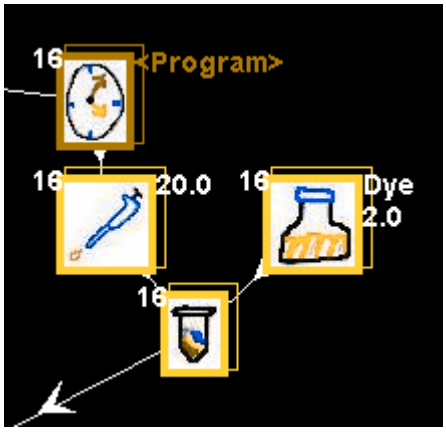

Figure 21: A color deficient vision simulation of Figure 20

22. **Individual icon information is not labeled**
Priority: *major (3)*
Heuristics violated: *recognition v. recall (6)*
Problem description: Tag numbers and volumes of individual icons, both numerical information, are not labeled, potentially causing a user to confuse the two.

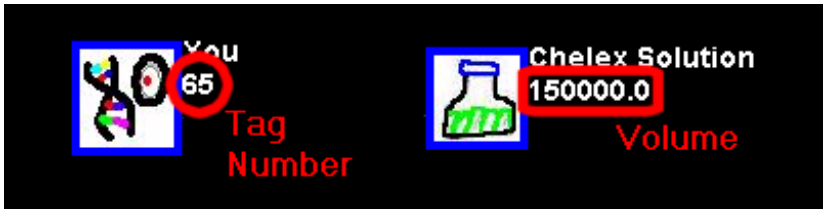

Figure 22: Tag numbers and volumes appear the same

23. **Volume measurements on icons lack units**
Priority: *major (3)*
Heuristics violated: *real-world match (2), recognition v. recall (6), flexibility (7)*
Problem description: The current unit-less system (Labscape does not track the units of measure in any way) forces the user to remember which units were used for reagent measurements.

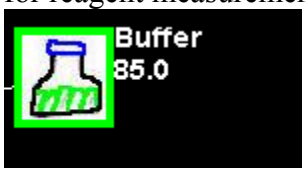

Figure 23: The volume text does not display the units of measure

24. **"Identical" icons have different sets of underlying attributes**
Priority: *minor (2)*
Heuristics violated: *consistency (4), recognition v. recall (6), error recovery (9)*
Problem description: Identical icons sometimes have different attributes. This violates the heuristic of "recognition vs. recall." If one icon can have more than one set of attributes, the user must remember each set of attributes for individual icons in the SFG.

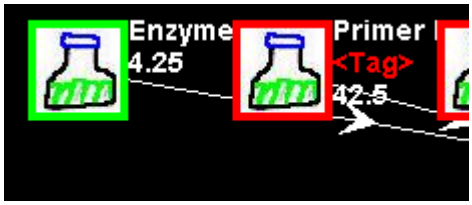

Figure 24: These two icons are identical, but have different label types

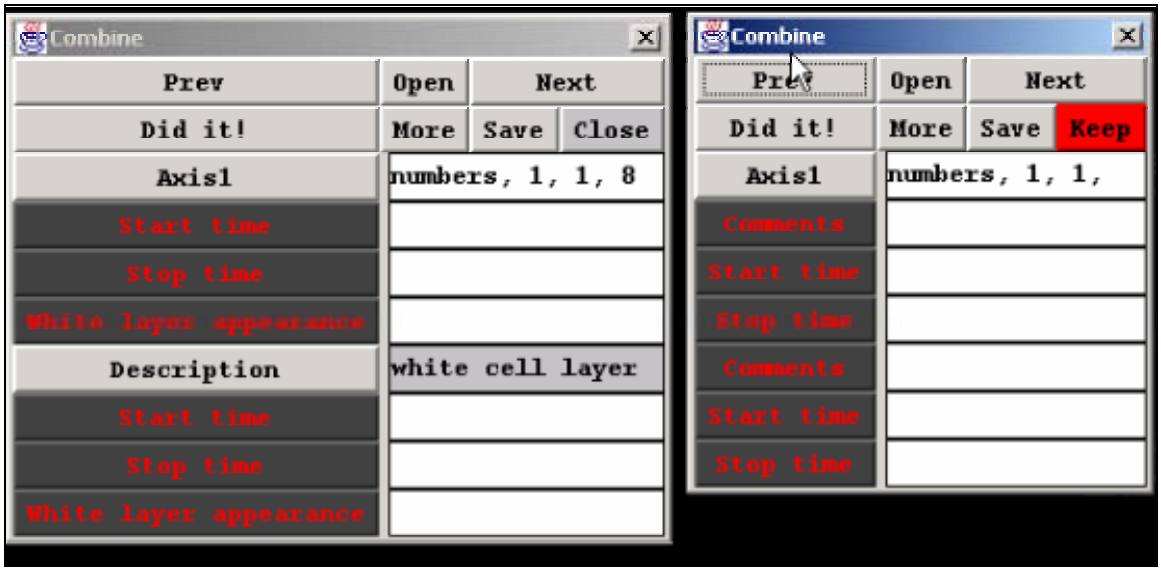

Figure 25: These two dialog windows are represented by the same icon (*combine*)

25. **Numbers in the icon labels do not use commas**
Priority: *cosmetic (1)*
Heuristics violated: *consistency (4), minimalist design (8)*
Problem description: The numbers that are listed to the right side of an icon do not display comma separators (e.g., *15000* instead of *15,000*). This makes the numbers slightly more difficult to parse, and therefore, more prone to errors (e.g., one too many or too few 0's).

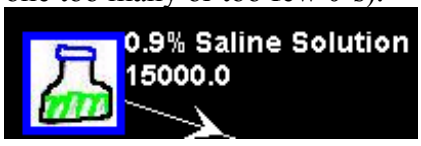

Figure 26: Large numbers lack comma separators

26. **Overlapping icons cannot be selected**
Priority: *major (3)*
Heuristics violated: *system status (1), flexibility (7), minimalist design (8), error recovery (9)*
Problem description: Icons can be dragged on top of other icons. Once this occurs, it is difficult to maneuver the icons. Icons are not stacked on top of each other in the order that they are moved. When the user attempts to select the icon on top, another icon in the stack may be selected instead.

Figure 27: A set of overlapping icons

27. **Labels can be obscured**
Priority: *minor (2)*
Heuristic violated: *system status (1)*
Problem description: Icons can be moved into positions where they block the text of another icon.

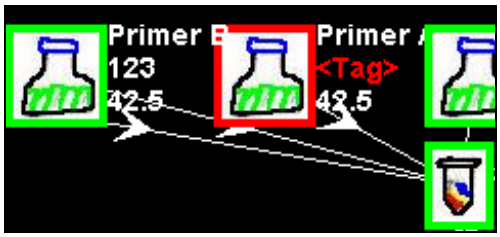

Figure 28: The text in these icons is obscured by neighboring icons

Dialog Windows

28. **Non-descriptive error messages**
Priority: *catastrophe (4)*
Heuristics violated: *error recovery (9)*
Problem description: Error messages give do not indicate which specific error has occurred. Without this information, it is difficult for the user to recover from the error.

Figure 29: An error message that fails to give specific information

29. ***Keep*, *Save* and *Did it!* buttons in dialog windows**
Priority: *major (3)*
Heuristics violated: *real-world match (2), consistency (4)*
Problem description: The difference between the *Keep*, *Save* and *Did it!* buttons in the dialog windows is ambiguous. When *Keep* is pressed, another dialog can be opened without the current window automatically closing, but any data that has not been saved is lost. When *Save* is pressed, the information in the dialog window is saved to the database and the window closes. When *Did it!* is pressed, the step is marked as completed, but any data that has not been saved is lost. These three terms are similar in meaning but perform three different actions in Labscape.

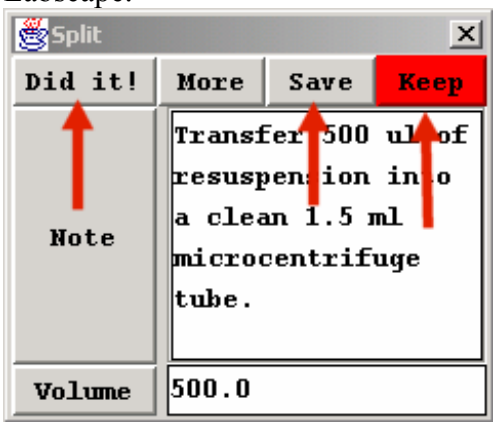

Figure 30: A dialog containing the *Save*, *Keep* and *Did it!* buttons

30. **No cancel button in dialog windows**
Priority: *minor (2)*
Heuristics violated: *consistency (4)*
Problem description: Without a cancel button, the user must remember to press the "x" in the upper right corner, or to press the *Keep* and *Close* buttons, to prevent a change in a dialog window from being saved.

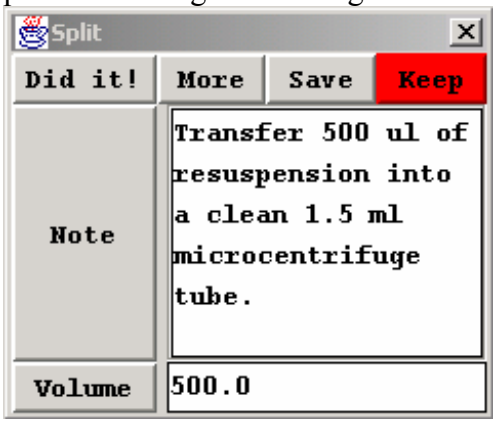

Figure 31: The top-right "x" is the only button that cancels changes and closes the window

31. **Dialog windows do not automatically update**
Priority: *minor (2)*
Heuristics violated: *consistency (4), error prevention (5)*
Problem description: The information entered into a dialog window is not automatically saved. If *Did it!*, *Keep*, or the "x" in the upper right corner of the window are pressed, all changes made to the window are lost unless the *Save* button has been pressed. This is inconsistent with the rest of Labscape, where all changes to the SFG are automatically saved.

32. **Several dialog windows have no title**
Priority: *major (3)*
Heuristics violated: *system status (1), consistency (4), minimalist design (8)*
Problem description: Without a title, it may be difficult for the user to remember the dialog window's function or purpose.

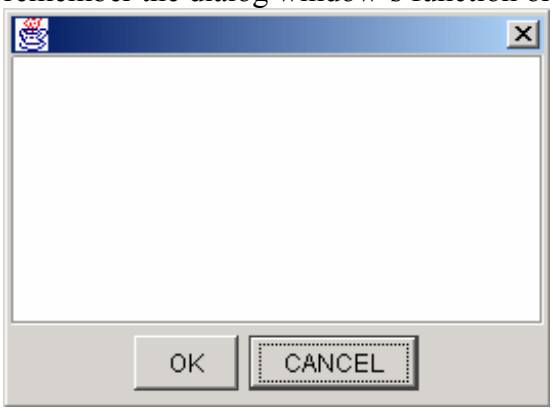

Figure 32: A window without a title

33. **Pop-up keyboard is in alphabetical order**
Priority: *cosmetic (1)*
Heuristics violated: *real-world match (2), consistency (4)*
Problem description: The pop-up keyboard has the appearance of a physical keyboard (QWERTY), yet has an alphabetical layout.

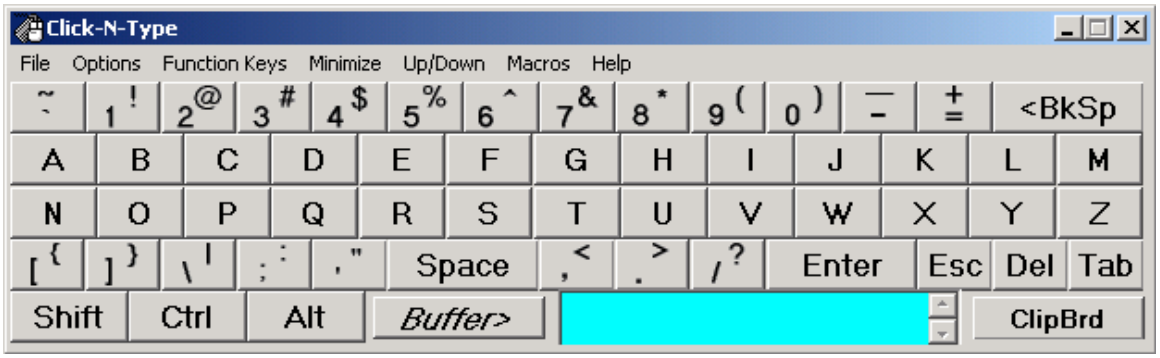

Figure 33: The pop-up keyboard supplied with Labscape

Figure 34: This visual keyboard (MS Windows *On-Screen Keyboard*) uses the QWERTY layout

34. **Error and informational messages have the same appearance**
Priority: *major (3)*
Heuristics violated: *consistency (4), recognition v. recall (6), error recovery (9)*
Problem description: The informational messages that the program generates are indistinguishable from the error messages. The user must carefully read the contents of the message to determine whether the message is telling the user to perform an action or diagnose an error.

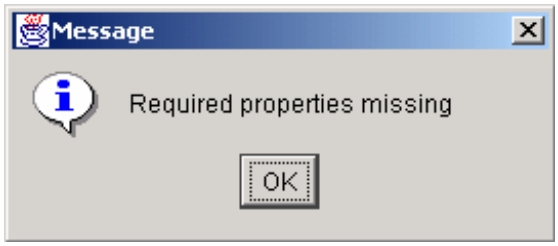

Figure 35: An error message

Figure 36: An informational message

35. **The *session selection window* is sometimes empty**
Priority: *major (3)*
Heuristics violated: *system status (1), error recovery (9)*
Problem description: When the *session selection window* appears, it sometimes contains no content.

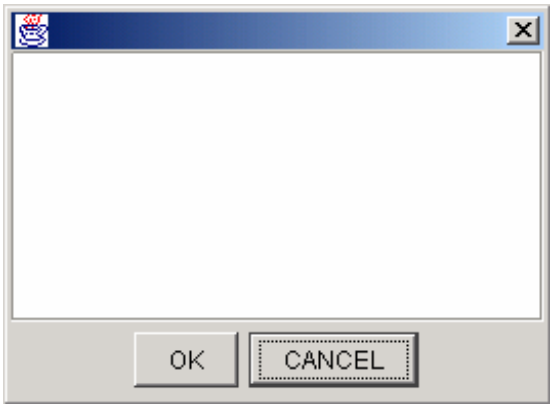

Figure 37: The *session selection window* contains no content

Pie Menu

36. **Unfamiliar terms**

a. Query

Priority: *major (3)*
Heuristics violated: *real-world match (2), consistency (4)*
Problem description: Words such as *query* may not be recognizable to users, as most programs with which they are familiar (e.g., Windows and Mac programs) use the word *search* to represent that function.

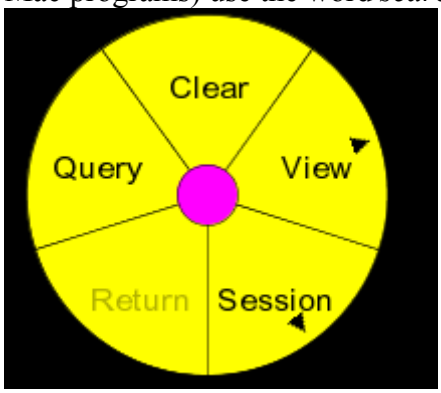

Figure 38: Terms on the pie menu may be unfamiliar to Labscape users

b. Clear

Priority: *major (3)*
Heuristics violated: *real-world match (2), consistency (4), error prevention (5)*
Problem description: Though the term *clear* may be familiar to users, exactly what gets cleared when the command is selected is ambiguous. When pressed, all of the dialog windows that are currently open are closed. However, users may believe that the *main window* or the data of the current dialog window will be cleared.

c. ***Return***
Priority: *major (3)*
Heuristic violated: *real-world match(2), consistency (4)*
Problem description: The *Return* option on the pie menu does not fully describe the action that it performs. It changes the user's view from that of a batch to the previous screen, the SFG. However, its function is not clear until the user has opened a batch and the option becomes available.

37. ***Clear* option available when no pop-ups are present**
Priority: *minor (2)*
Heuristics violated: *system status (1), consistency (4)*
Problem description: The *clear* option is available in the pie menu when no pop-up windows are open. This misrepresents the status of the system to the user. The user may think that because the option remains available, it may do more than close any open dialog windows.

38. **Center of pie menu provides no indication that it closes the menu**
Priority: *minor (2)*
Heuristics violated: *system status (1), consistency (4)*
Problem description: The pop-up pie menu is closed by clicking on the center circle. However, there is no indication that clicking on the center circle will close the menu. Compounding the problem, target users for Labscape are familiar with Windows and some Mac software, and will likely be unfamiliar with pie menus.

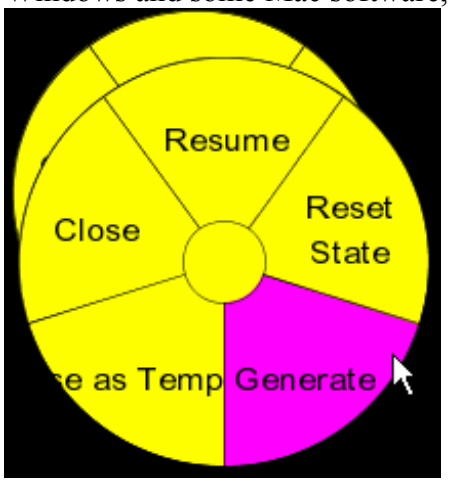

Figure 39: The center of the pie menu closes it

39. **Inconsistent pie menu coloring on the *View* sub-menu**
Priority: *major (3)*
Heuristic violated: *consistency (4)*
Problem description: When the user clicks on *View* from the pie menu, the resulting sub-menu appears in shades of gray, as opposed to the yellow and magenta colors seen in other menus. Users may be confused by this, as gray is often used to indicate an option is not available.

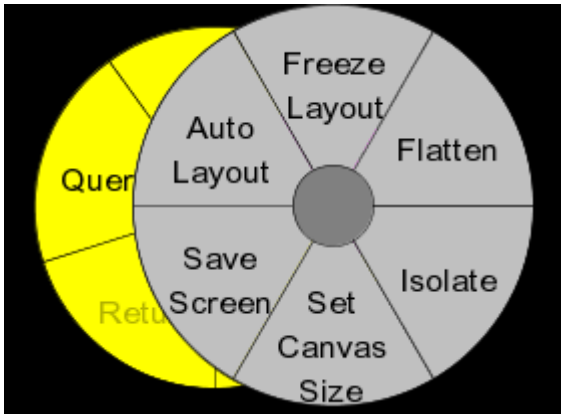

Figure 40: Inconsistent coloring on the *View* menu

40. **Inactive pie layer appears active**
Priority: *cosmetic (1)*
Heuristic violated: *system status (1)*
Problem description: When a pie sub-menu is opened, the original pie menu is obscured, but still visible. Because no visual change to the first-level pie menu occurred (other than being obscured), a user may attempt to click on an option on the background pie menu because it appears to be clickable, even though it is inactive.

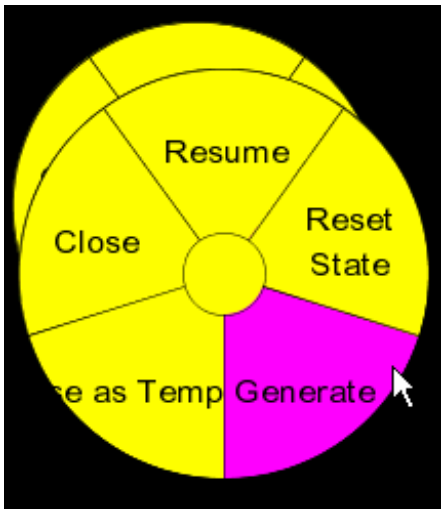

Figure 41: The background pie appears active based upon its color

41. **Black text on a magenta background is difficult to read**
Priority: *minor (2)*
Heuristics violated: *system status (1), consistency (4), error prevention (5), minimalist design (8)*
Problem description: On the pie menu, the system indicates which portion of the pie is active by highlighting the area in magenta. However, the magenta highlight lowers the contrast between the text and the background, making the text harder to read. On the other hand, the contrast between the yellow

background and the black text is higher, drawing the eye to the portion of the menu that is not highlighted.

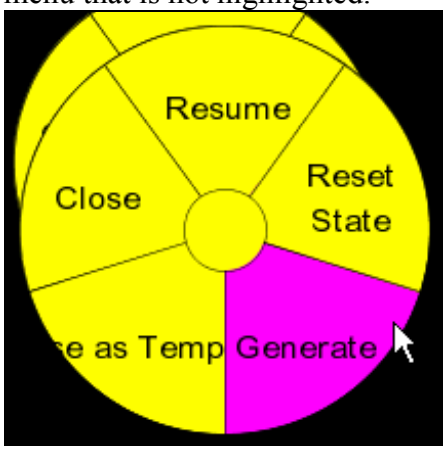

Figure 42: The highlighted pie section is black text on magenta

42. **Functions can only be accessed by pie menu**
Priority: *minor (2)*
Heuristics violated: *system status (1), flexibility (7)*
Problem description: In order for a user to access a Labscape function (e.g., close the session, open an existing session, create a new session), the pie menu must be used. There are no alternatives for users who are either not familiar with pie menus or who are advanced enough that a pie menu slows down their work. Icons can be clicked to bring up input dialogs; however, the pie menu is the only way to access general Labscape functions.

43. **Pie menu can be obscured**
Priority: *cosmetic (1)*
Heuristics violated: *flexibility (7), error prevention (5)*
Problem description: The pie menu can be opened in areas where it is partially obscured by the borders of the window.

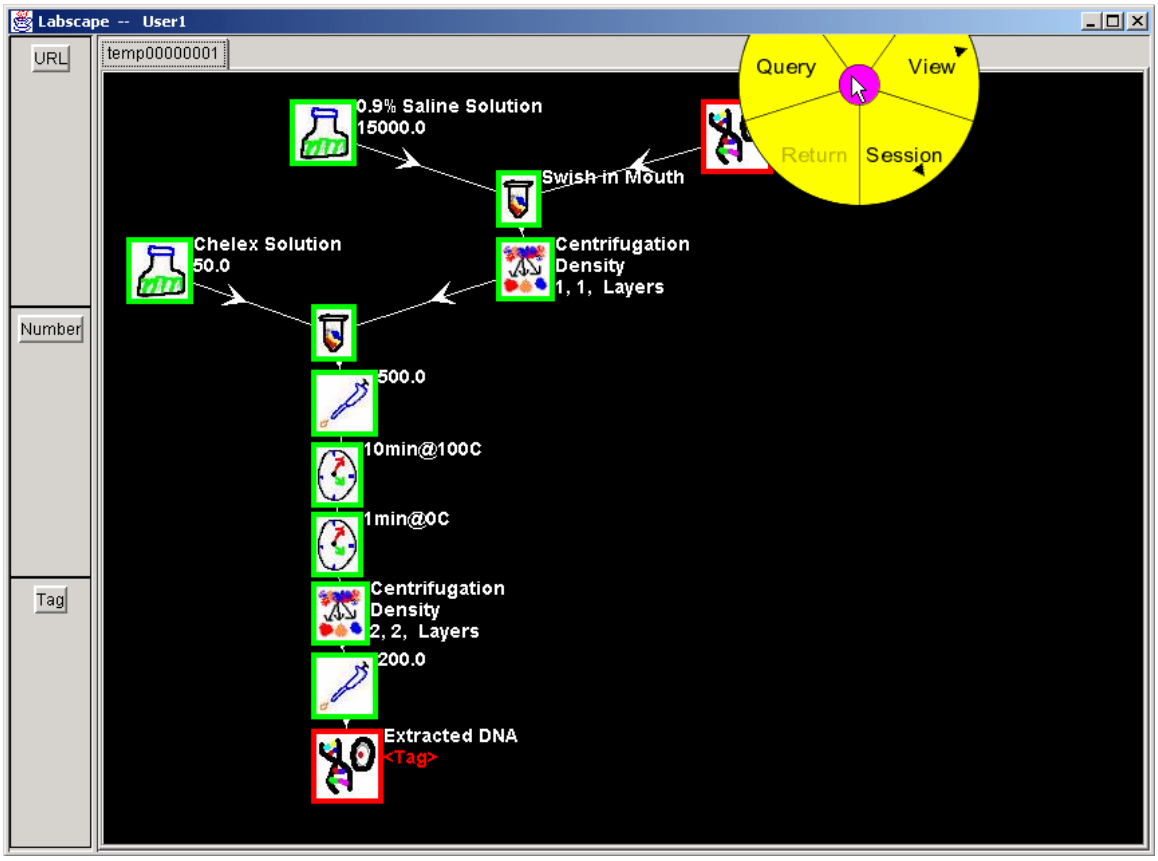

Figure 43: The pie menu can be obscured by the window frame

44. **Debugging tools appear as normal Labscape features**
Priority: *major (3)*
Heuristics violated: *real-world match (2), error prevention (5), error recovery (9)*
Problem description: The *Reset State* option on the *Session* pie menu (see Fig. 44) is a debugging tool for developers; it is not meant to be a feature for users. Despite that fact, the debugging tool is not only co-located with other normal Labscape functions, it also has the same appearance.

45. **Pie menu text extends beyond border**
Priority: *major (3)*
Heuristics violated: *error prevention (5), minimalist design (8), consistency (4)*
Problem description: The text of long menu items extends beyond the border of the pie menu. Because both the background of the main session window and menu text are black, (e.g., *...as Temp* in *Session* pie menu; see Fig. 44) this text is impossible to read.

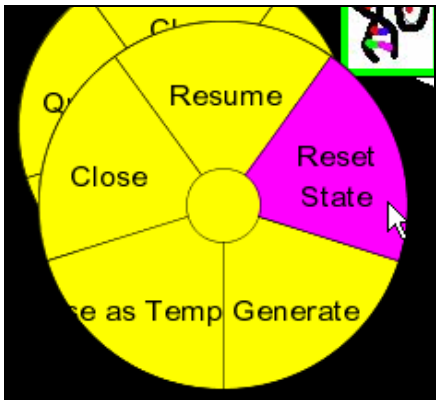

Figure 44: The text in the pie menu extends past the edge of the pie

46. **Lack of indication that a user must click to view pie menu**
Priority: *minor (2)*
Heuristic violated: *system status (1)*
Problem description: There is no indication from the system that it is accepting commands until the pie menu has been brought up. On Windows and Macintosh systems, menu bars are almost always visible. For example, *File* and *Edit* may always be visible in a given application. However, in Labscape, there is no visual cue that informs the user that clicking on an empty part of the *main window* will bring up a menu.

Functionality

47. **Lack of security**
Priority: *catastrophe (4)*
Heuristic violated: *consistency (4)*
Problem description: Labscape currently has no security mechanisms. Anyone with access to Labscape can log in as a listed user and alter experiments and/or data without leaving any record of changes made or who made the changes.

48. **No "redo" functionality**
Priority: *minor (2)*
Heuristics violated: *user control (3), consistency (4), error prevention (5)*
Problem description: After a user has pressed *undo*, there is no way to restore the previous state of the session without manually changing the data.

49. ***Undo* does not "undo" last change**
Priority: *major (3)*
Heuristics violated: *real-world match (2), user control (3), consistency (4), error prevention (5)*
Problem description: In most Windows and Mac software (the software with which Labscape users are most familiar), "undo" means to reverse the last change made by the user. In Labscape, *undo* means to mark a step in the experiment as "not completed."

50. **Duplicate session names**
Priority: *catastrophe (4)*
Heuristics violated: *system status (1), real-world match (2), user control (3), consistency (4), error prevention (5), recognition v. recall (6), error recovery (9)*
Problem description: When a user attempts to create a new session that has a name identical to an existing session, Labscape does not notify the user that a session already exists with that name. Without this information, a user could continue working in the session he or she was last using without realizing that the creation of the new session failed. For example, if User A creates a session called *session1* and begins working, and User B also creates a session with that name, Labscape will store the information for both experiments in the same location. Neither user will know that an error has occurred until one of them closes *session1* and then resumes it, only to find that the data for both experiments has been stored in a single session.

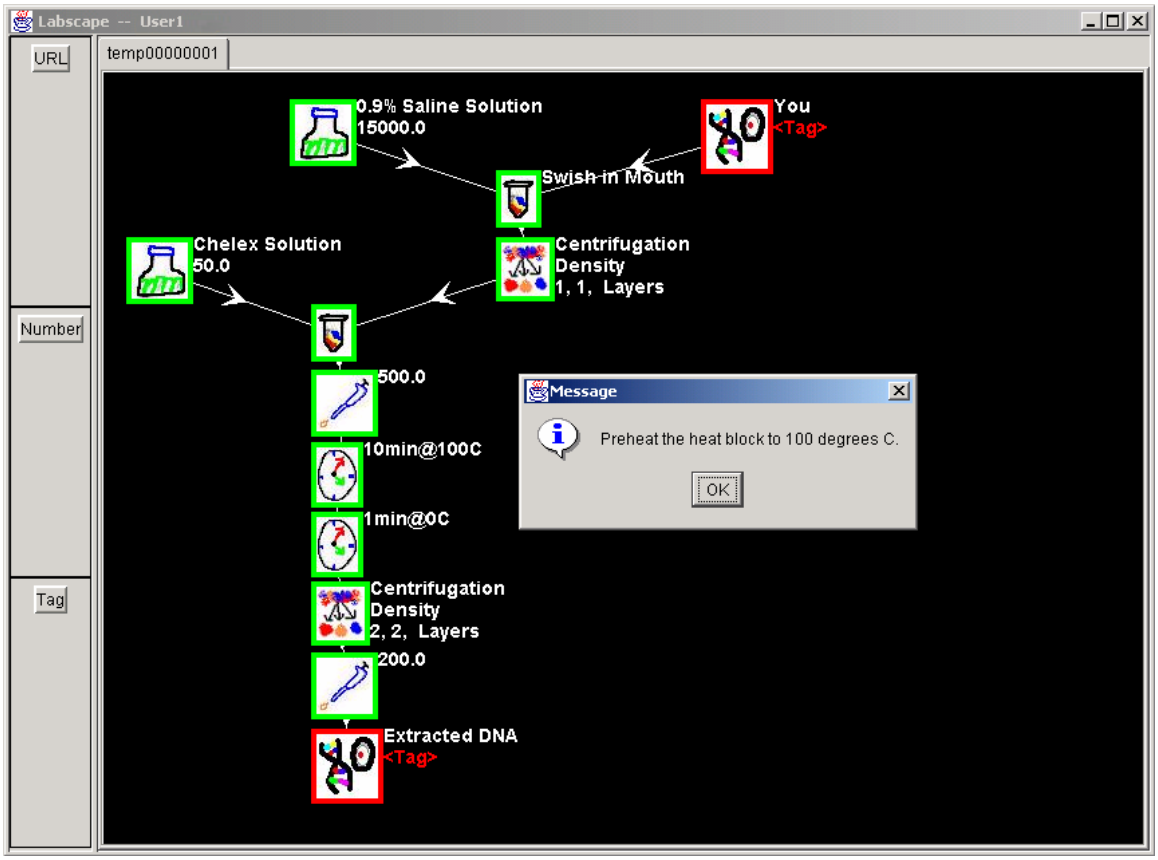

Figure 45: Labscape behaves as if no error occurred (by giving information on what to do in the new experiment) when a new session is given the same name as an existing session

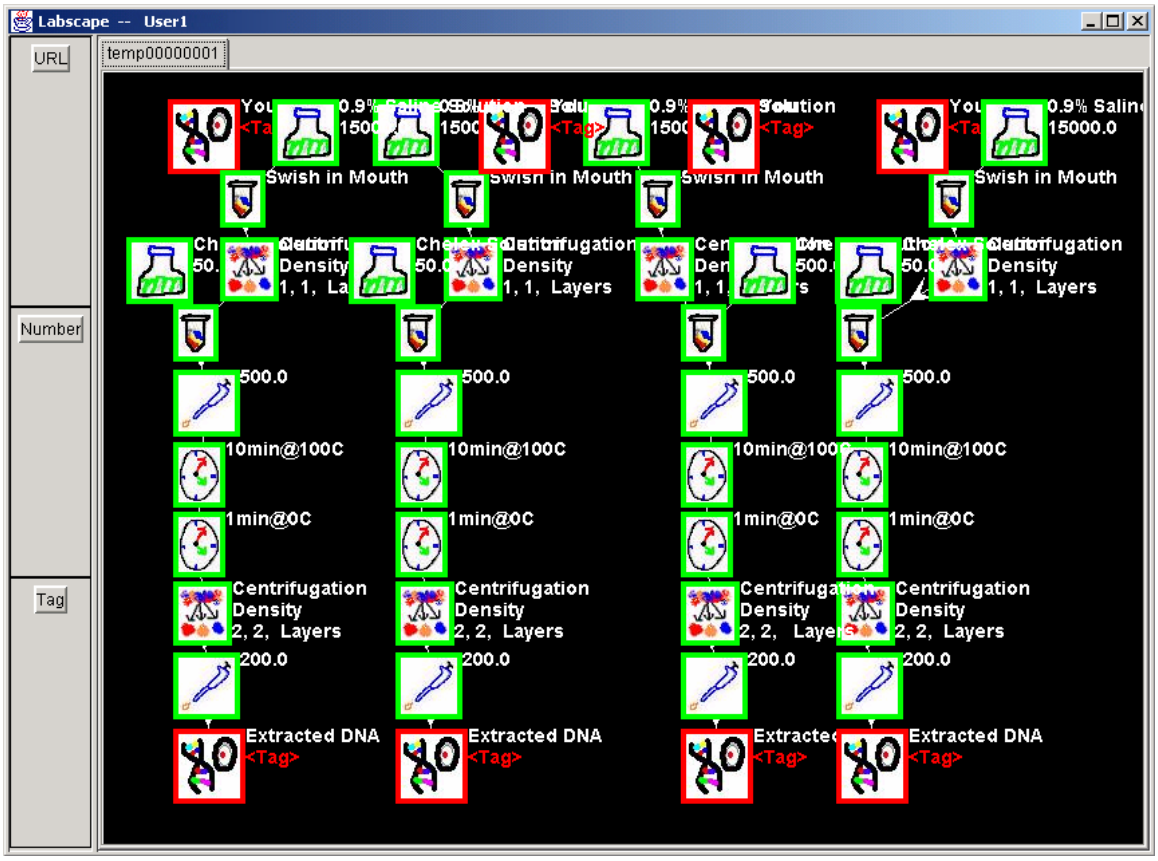

Figure 46: The result of creating four identical sessions with the same name

51. **Handling of invalid input**
Priority: *major (3)*
Heuristics violated: *consistency (4), error prevention (5), minimalist design (8)*
Problem description: If an invalid data string is entered in a reagent dialog, Labscape does not notify the user. Instead, the invalid data string is ignored and Labscape keeps the last valid value entered.

52. **No menu item to close Labscape**
Priority: *minor (2)*
Heuristics violated: *system status (1), user control (3), consistency (4)*
Problem description: The only way to quit Labscape is by clicking on the "x" in the corner of the main window. This is inconsistent with other software products with which the target user group is familiar (e.g., Microsoft Excel) that use a menu option (e.g., Exit) to allow the user to close the software.

53. **No indication given that Labscape is contacting the database**
Priority: *minor (2)*
Heuristic violated: *system status (1)*
Problem description: When the user tries to open a new session, Labscape does not notify the user that it is connecting to the database to retrieve a list of the possible sessions that may be opened. If the database connection is slow, the user

may not know why the *session selection window* does not appear. The user may then attempt to open an additional session. Because two session list requests can be made simultaneously, a dialog window will be supplied for each time the list is requested.

Global Comments

54. **Use of *Tag***
Priority: *cosmetic (1)*
Heuristics violated: *real-world match (2)*
Problem description: The word *tag* (in Labscape, meaning the label placed on a specific sample or reagent) may differ from the terminology used by the members of the target user groups. (NOTE: users should be consulted to verify this before any changes are made).

55. **Use of the color red is inconsistent**
Priority: *major (3)*
Heuristic violated: *consistency (4), error prevention (9)*
Problem description: The color red is used to represent different things in Labscape. In dialog windows, the *Keep* button is red for emphasis, and red is also used to represent missing information; on the main session window, a red outline around an icon represents state.

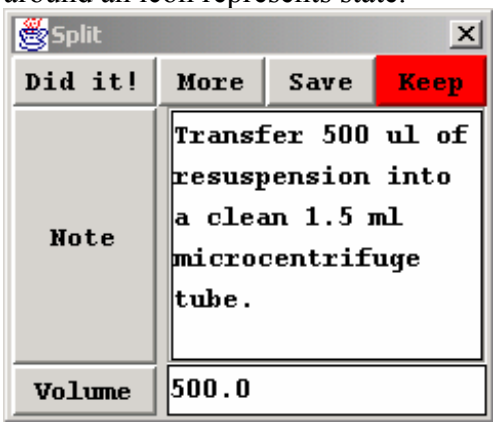

Figure 47: The *Keep* button is red for emphasis

Figure 48: In this case, red indicates missing information

Next Steps

This report will be used by the design/development team to facilitate a redesign of the Labscape UI. The authors of this evaluation will meet with the design/development team to discuss the problems that were found. This meeting will allow both teams to provide input as to which problems are the most important to fix and which are less important. It is likely that it will not be feasible to fix all the problems identified in this evaluation.

Acknowledgements

The authors would like to thank Jeong Kim for her help with the editing of this paper, Jing Su for his assistance in guiding us through the Labscape UI, and Larry Arnstein for his support and the information he provided to this project.

Appendix A: About Heuristic Evaluations

Heuristic evaluations are used to identify defects with the user interface of an application or design [Nielsen, 1994]. This usability inspection method was developed by Jakob Nielsen as a way to quickly and economically test interfaces. Traditional usability studies can be time consuming and expensive; this is because actual users are used to evaluate the application. A heuristic evaluation is quick and inexpensive because the evaluators need not be members of the application's user group. Another advantage is that it may be used on a system in development.

A heuristic evaluation is also a good tool to prepare a UI for a study with actual users. It ensures that heuristic violations are fixed before the users see the UI so that user testing will uncover usability problems that heuristic evaluations are not designed to detect.

Heuristic evaluations are not used to pinpoint the successes of a user interface. This means that a heuristic evaluation report will not include positive feedback; the evaluation will only identify the shortcomings of the user interface [Web-creators, 2001]. Heuristic evaluations are not a complete review of the interface because they only identify the heuristics violated; some usability problems may be overlooked.

Furthermore, a heuristic evaluation is not a user study or a task walkthrough. As stated earlier, the evaluators are not necessarily the users of the product. The evaluators are only looking for heuristic violations, so it is not necessary to place them in the position of a user performing a task. The list of heuristics is assumed to be a set of universal usability requirements that are independent of the user group.

Purpose

The purpose of most heuristic evaluations is to locate usability defects in a user interface (UI) that is under development.

How does a heuristic evaluation work?

Generally speaking, a group of three to five evaluators is brought together to evaluate the UI of a system. Each evaluator is given a list of heuristics (see Appendix B) that will be used to evaluate the UI and classify the usability defects. During the evaluation session, the evaluators will ideally view the entire UI at least twice.

The first pass through the interface is designed for the evaluators to gain familiarity with the application and to take preliminary notes on potential usability problems. On the second pass, evaluators look at each dialog in the UI and compare it against the list of heuristics. If an evaluator notices that some aspect of the UI violates a heuristic, it is noted and the evaluation continues. One evaluator takes the role of *moderator* and another takes the role of *recorder*. The moderator's job is to enforce the rules of the evaluation (Appendix C) and to ensure that the session progresses smoothly. The

recorder's job is to listen to the evaluators' discussion and to compile a list of the usability defects they find.

During heuristic evaluations, the evaluators must follow specific rules. These rules (see Appendix C) are designed to ensure that the evaluators focus on *identifying* usability problems, instead of discussing solutions or debating whether or not something is a flaw.

At the end of the evaluation, a report (such as this one) is compiled containing a list of the usability problems found, along with screenshots and problem descriptions that can be used by the designers/developers to locate the problems found by the evaluators.

What information can a heuristic evaluation provide?

The result of a heuristic evaluation is essentially a list of usability problems that the evaluators have located. Each item in the list contains a description of the problem, an explanation of why it is a problem and an indication of how serious the problem is from a usability standpoint. It is worth noting that usability problems that do not violate any of the heuristics are not identified in heuristic evaluations.

This information can be used by the designers/developers of the project in a number of ways. Because the heuristic evaluation produces a list of defects and their severity, the designer/developer may use this information in order to determine which aspects of the design should be fixed first. Furthermore, the resources may not exist to address each problem, forcing the designer/developer to make tradeoffs. The information provided by the heuristic evaluation can enable a designer/developer to make informed decisions when selecting which tradeoffs must be made. The information can also be used when designing usability studies if the evaluators wish to improve (or avoid) certain areas.

Appendix B: The heuristics used

[Neilsen and Mack, 1994, p.30]

1. **Visibility of system status** - The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
2. **Match between system and the real world** - The system should speak the user's language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
3. **User control and freedom** - Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.
4. **Consistency and standards** - Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
5. **Error prevention** - Even better than good error messages is a careful design which prevents a problem from occurring in the first place.
6. **Recognition rather than recall** - Make objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
7. **Flexibility and efficiency of use** - Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
8. **Aesthetic and minimalist design** - Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
9. **Help users recognize, diagnose, and recover from errors** - Error messages should be expressed in plain language (no codes), precisely indicate the problem and constructively suggest a solution.
10. **Help and documentation** - Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out and not be too large.

Appendix C: Rules of the heuristic evaluation

[Web-creators, 2001]

- **Identify defects and inconsistencies**: The job of the evaluator is not to assess the interface of the whole. Inconsistencies and defects are the focus of this type of evaluation. Successes of the interface will not be noted.
- **No explanations or excuses**: The evaluators should not try to defend the current design if there are potential usability problems. The problem should be noted for later discussion.
- **No debate, note and move on**: Every suggestion should be recorded so that discussion can take place at a later time.
- **Do not redesign**: The evaluators should not discuss possible solutions to the usability problem, as this takes time and does not contribute to the overall purpose of the heuristic evaluation.
- **No decisions or promises**: No decisions on how changes will be made should take place. Furthermore, promises that changes will be made should not be a part of the evaluation.
- **Focus on heuristics, not personal taste:** A heuristic evaluation is based upon a set of agreed-upon heuristics, not the personal aesthetic tastes of the evaluators. Basing the evaluation upon personal taste biases the results and may impede the evaluation procedure.

Appendix D: Severity rating scale for priorities

[Neilsen and Mack, 1994, p.49]

0: I don't agree that this is a usability problem at all
1: Cosmetic problem only—need not be fixed unless extra time is available on project
2: Minor usability problem—fixing this should be given low priority
3: Major usability problem—important to fix, so should be given high priority
4: Usability catastrophe—imperative to fix this before product can be released

Appendix E: Figure index

Appendix F: UI screenshots

Following are screenshots of how the UI appeared for this heuristic evaluation.

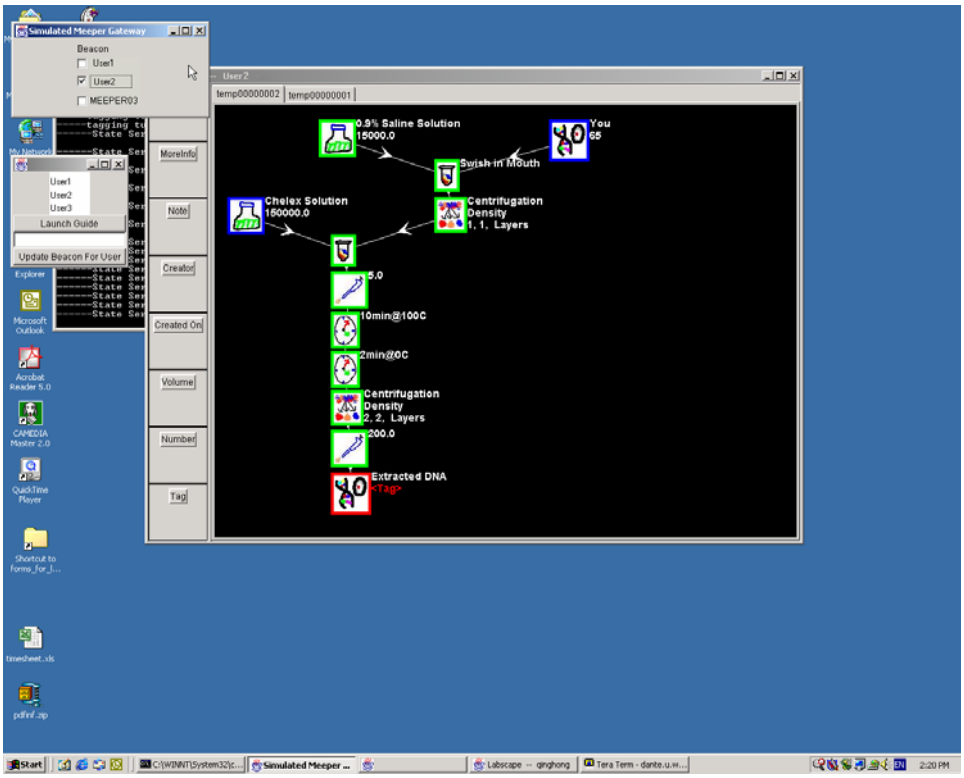

Figure 49: The *user check-in window*, *debug window* and *main window*

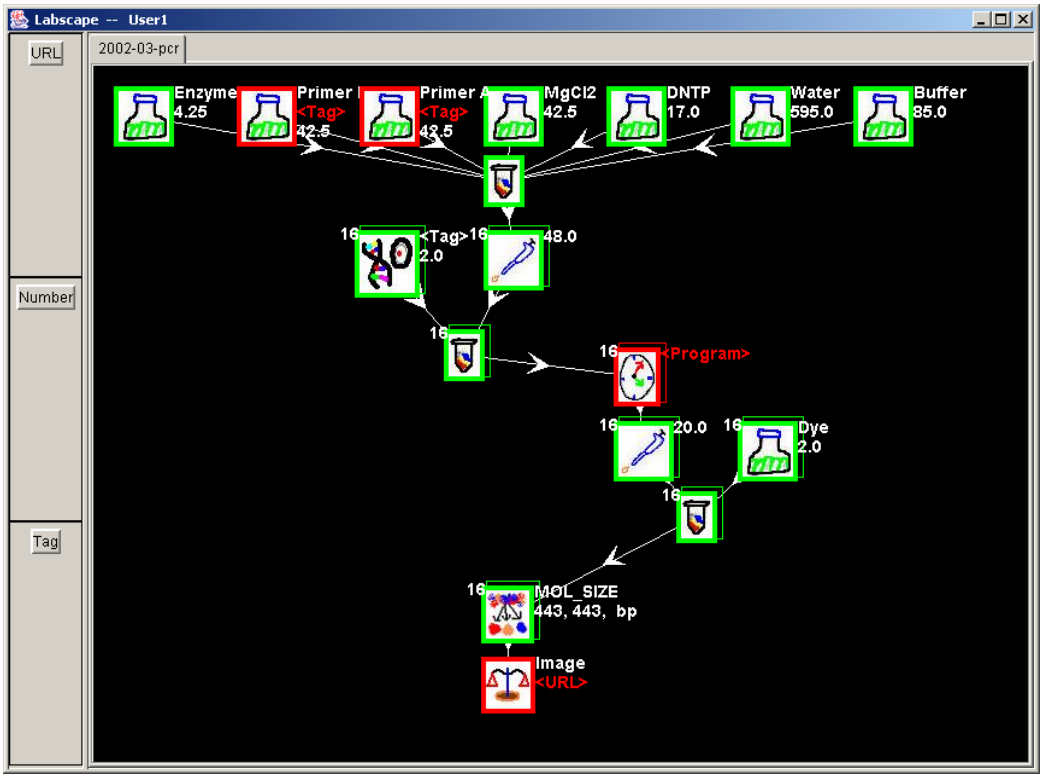

Figure 50: The *sample flow graph* in the *main window*

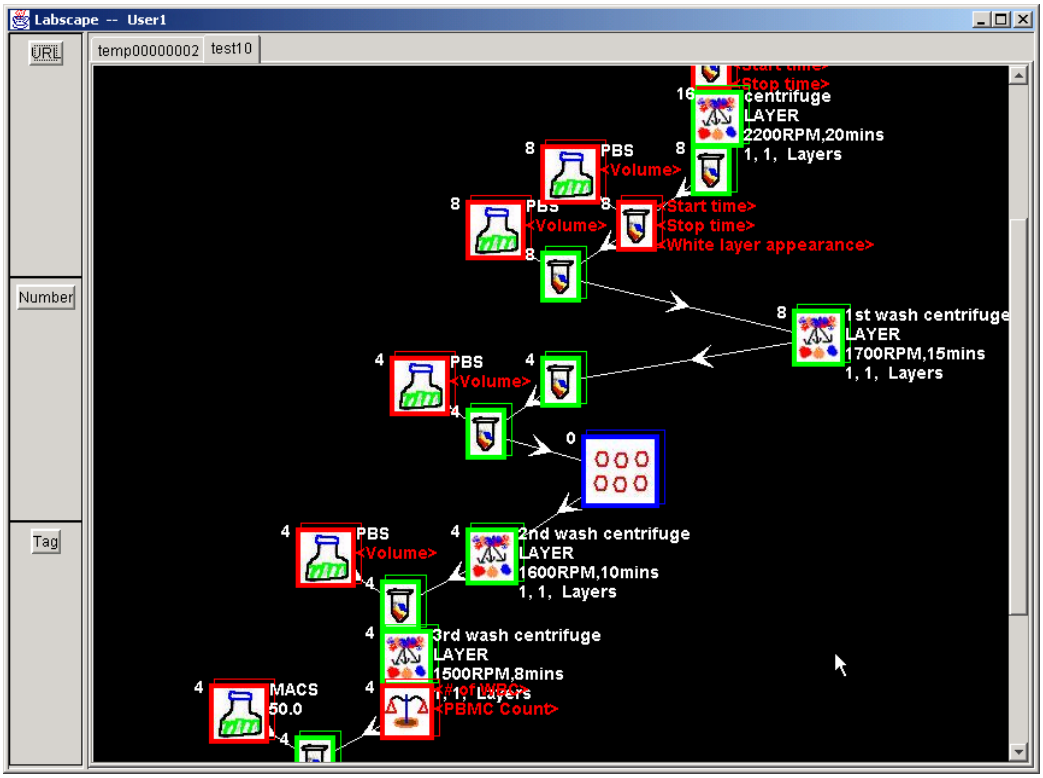

Figure 51: A *main window* utilizing a scrollbar

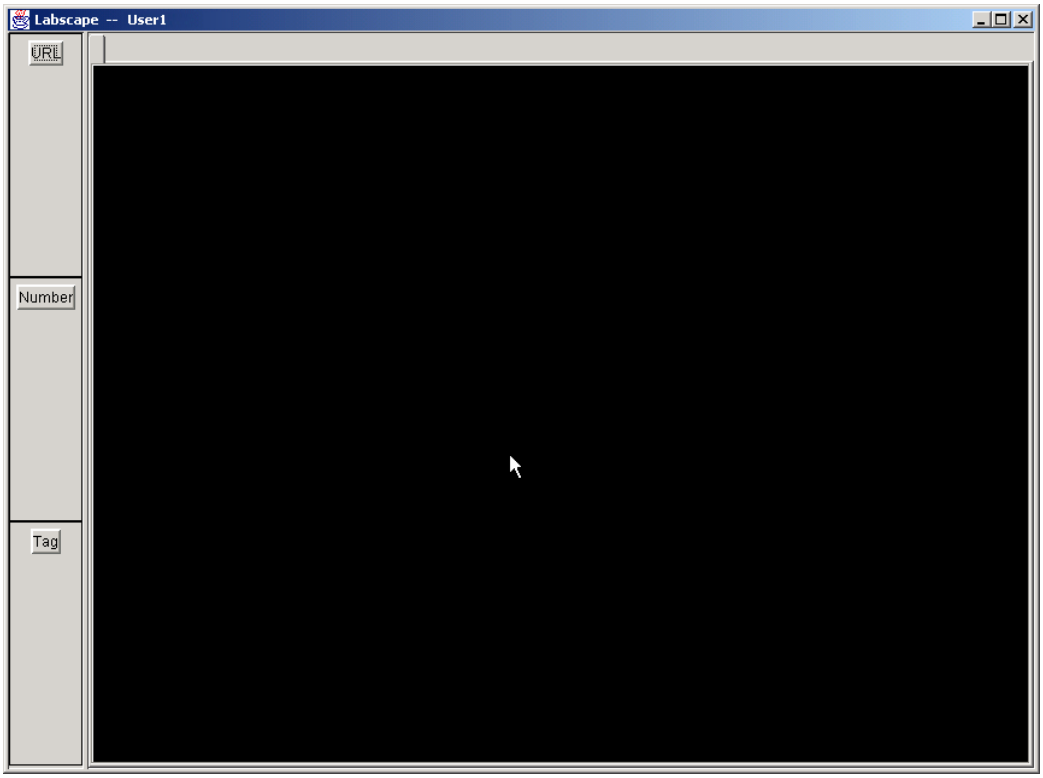

Figure 52: Initial appearance of the *main window*

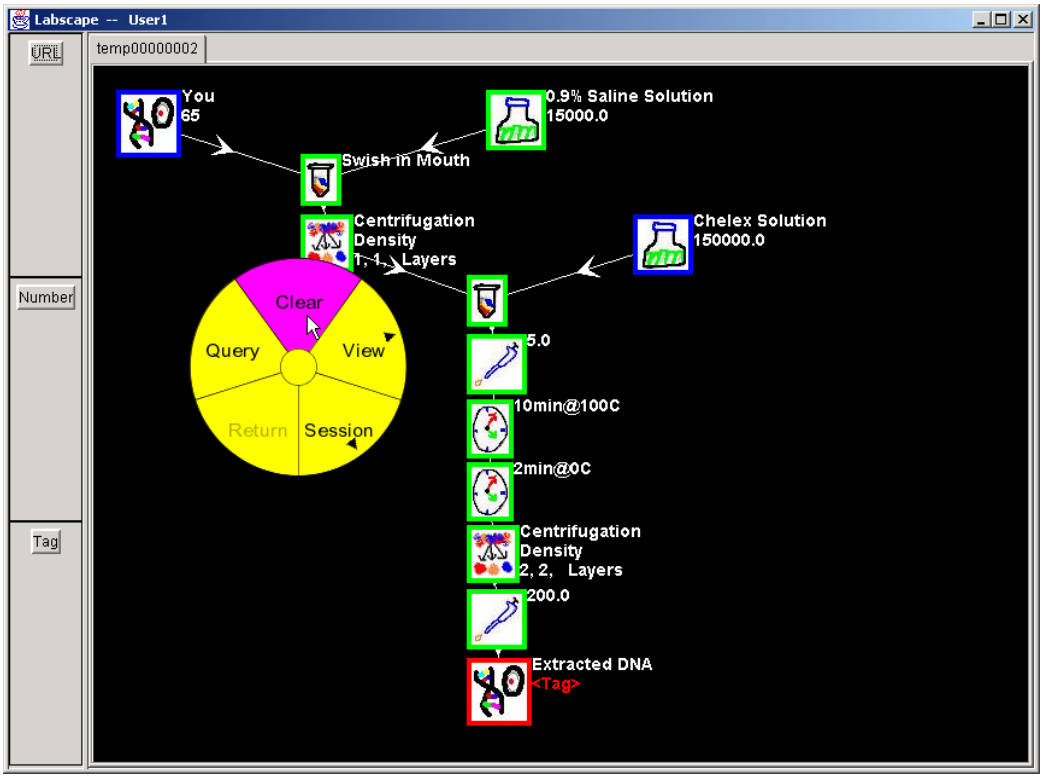

Figure 53: The primary pie menu

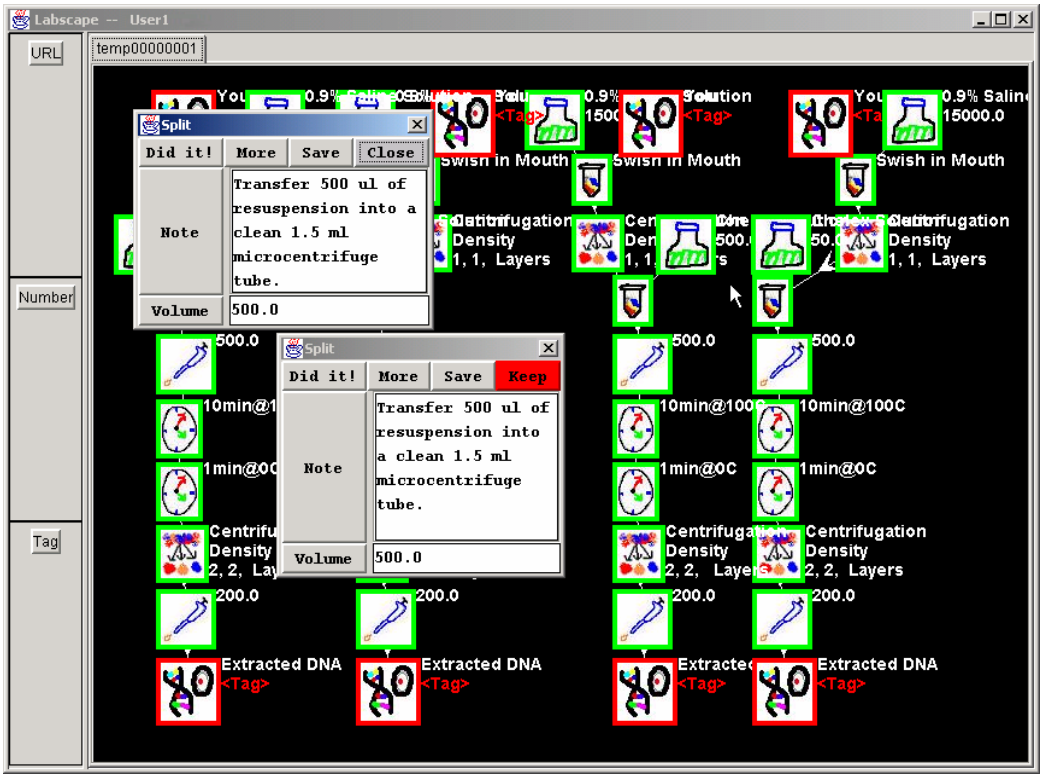

Figure 54: Two dialog windows open using *Keep*

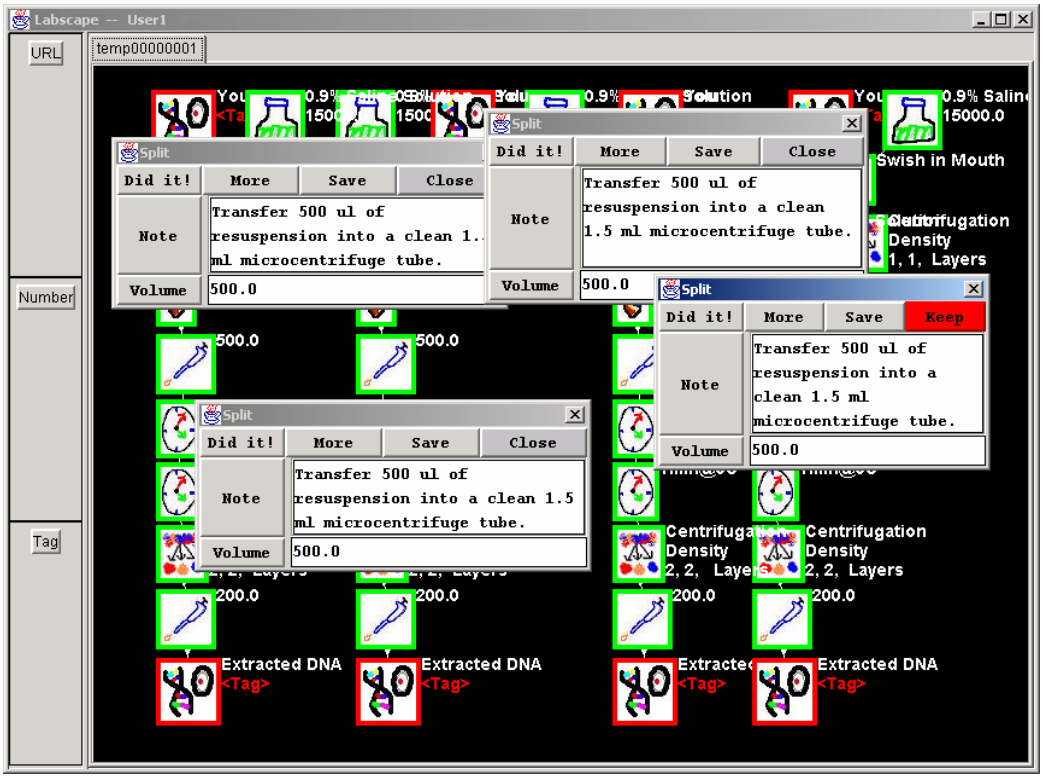

Figure 55: Multiple dialog windows open

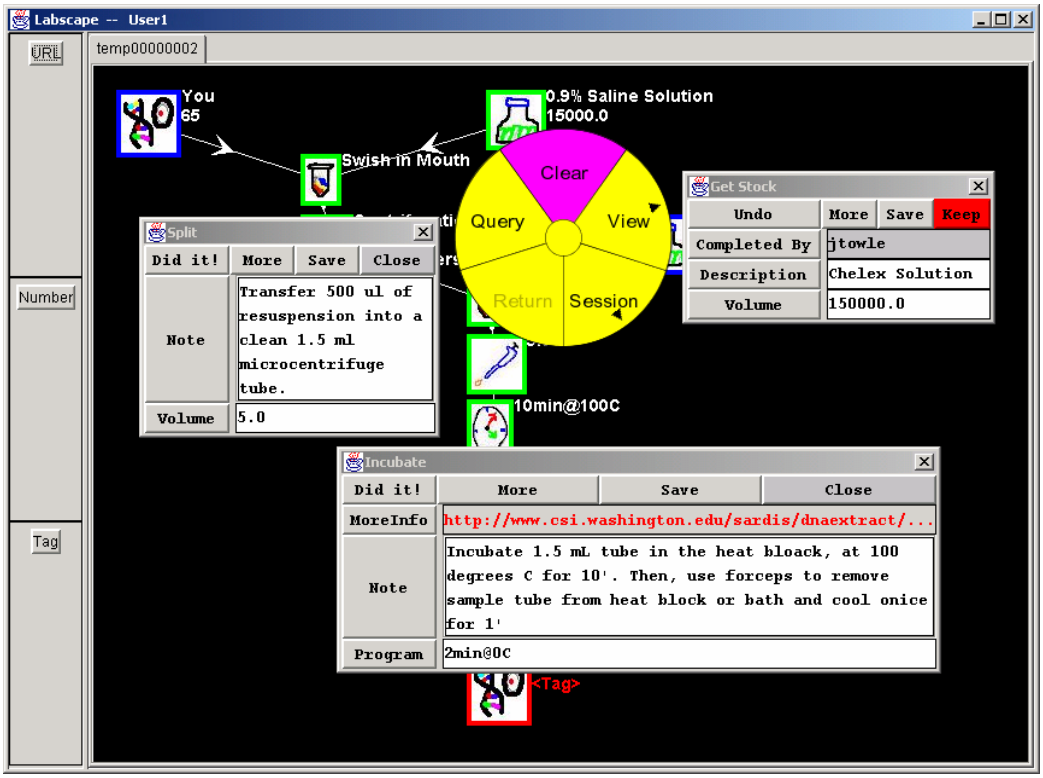

Figure 56: Use of the *clear* option

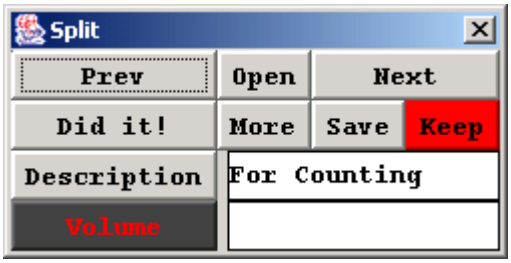

Figure 57: *Split* dialog window (from *Dispense* icon)

Figure 58: *Store Sample* dialog window (from *Detect* or *Biological Sample* icon)

Figure 59: *URLMeasure* dialog window (from *Detect* icon)

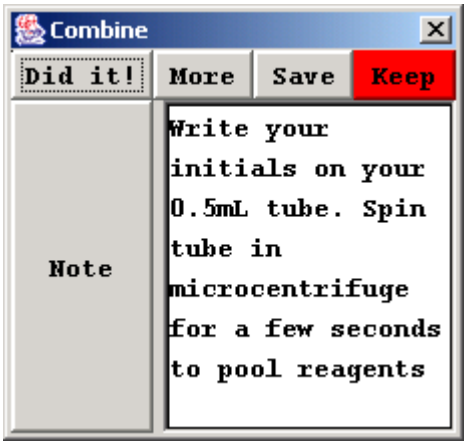

Figure 60: *Combine* dialog windows (from *Combine* icon)

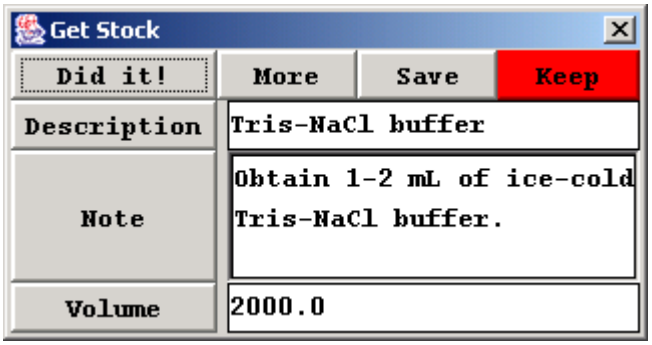

Figure 61: *Get Stock* dialog (from *Reagent* icon)

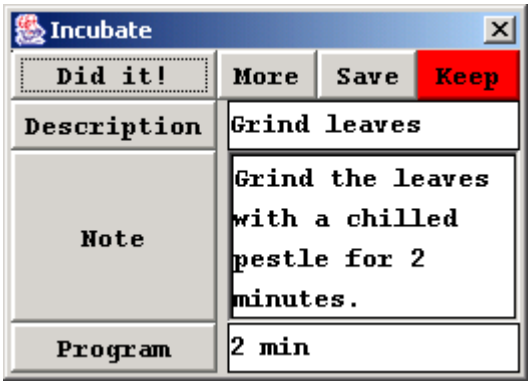

Figure 62: *Incubate* dialog window (from *Incubate* icon)

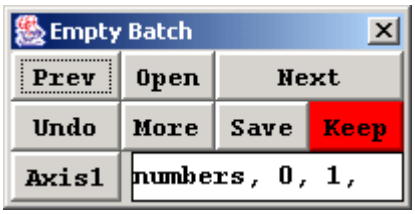

Figure 63: *Empty Batch* dialog window (from *Generic Batch* icon)

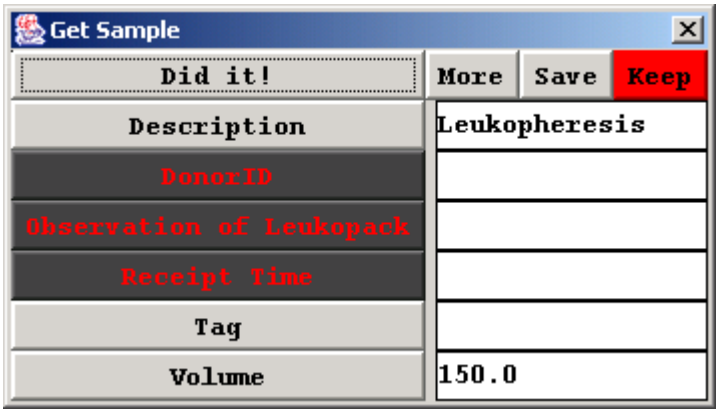

Figure 64: *Get Sample* dialog window (from *Biological Sample* icon)

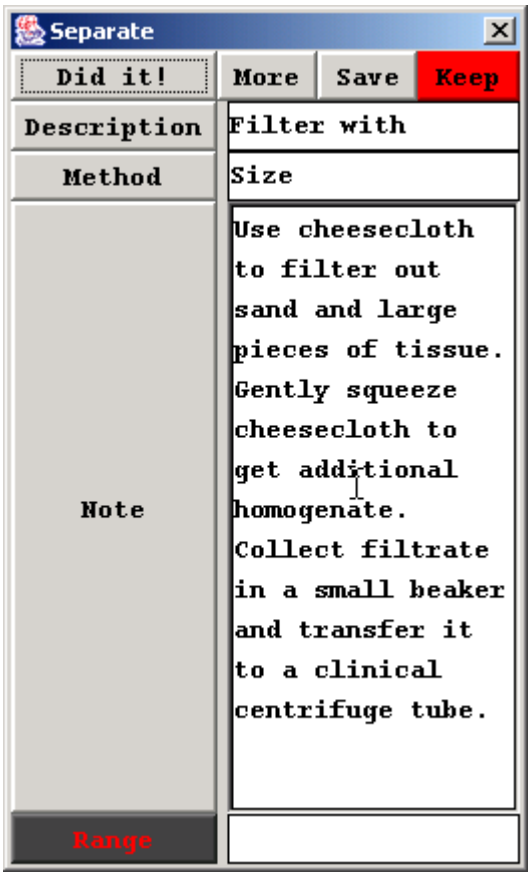

Figure 65: *Separate* dialog window (from *Separate* icon)

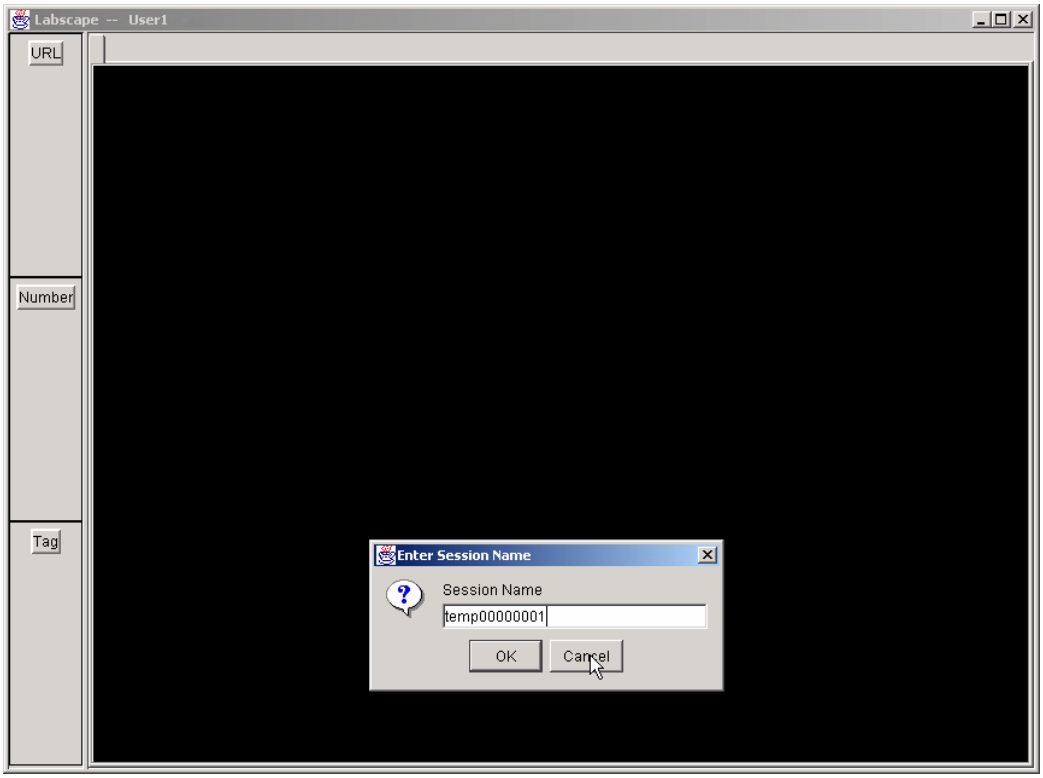

Figure 66: Creating a new session

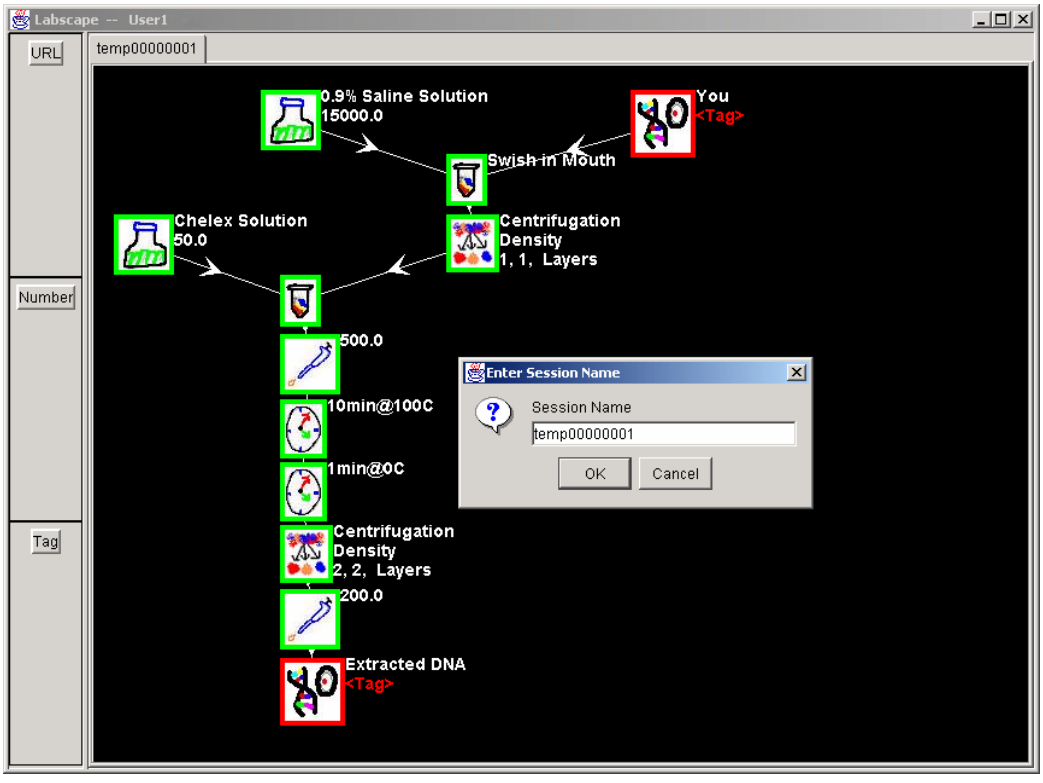

Figure 67: Naming a new session with an existing name (see Fig. 66)

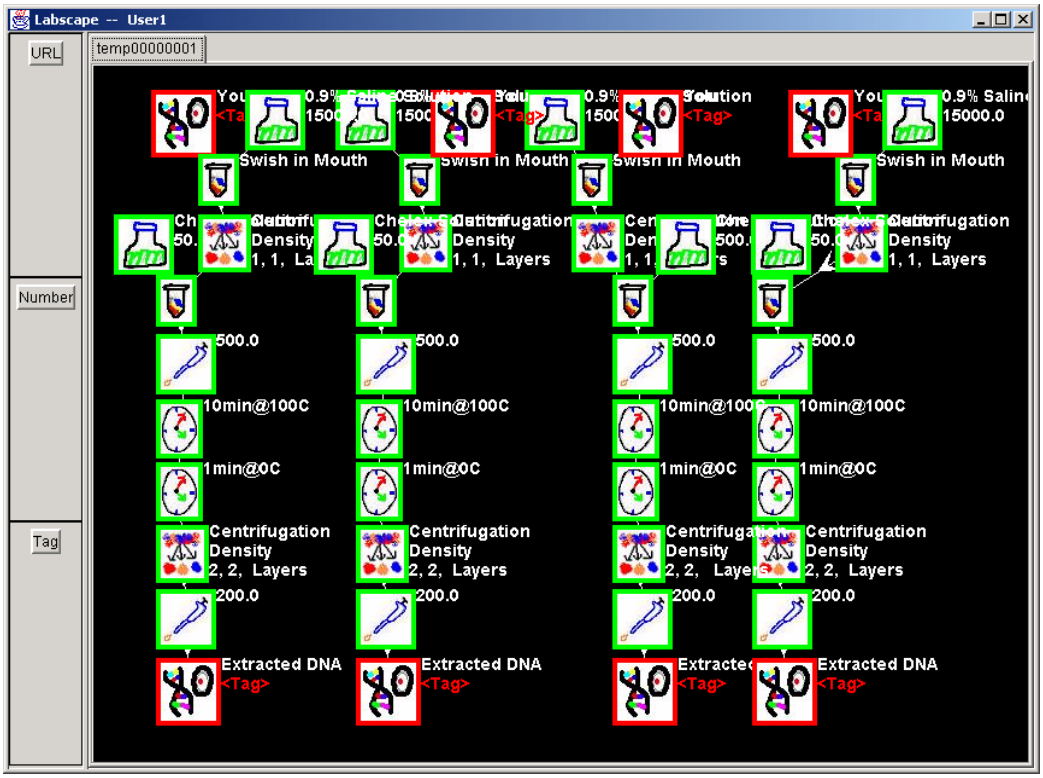

Figure 68: The result of creating multiple sessions with the same name

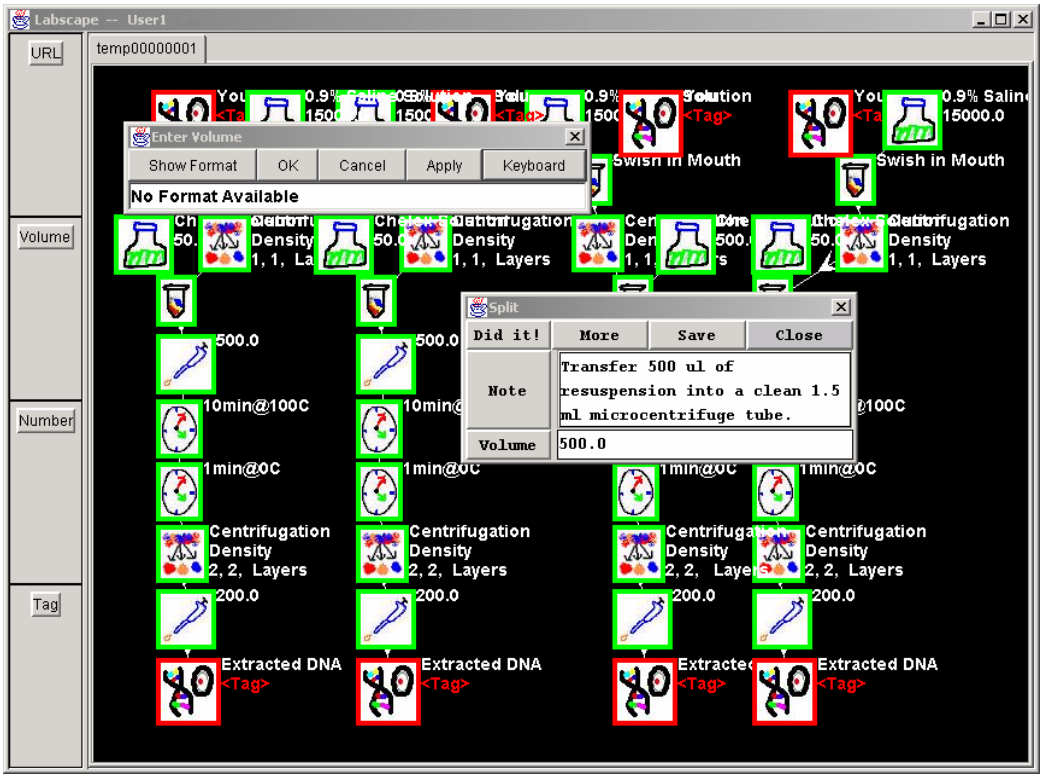

Figure 69: Formatted data entry dialog (w/ keyboard button)

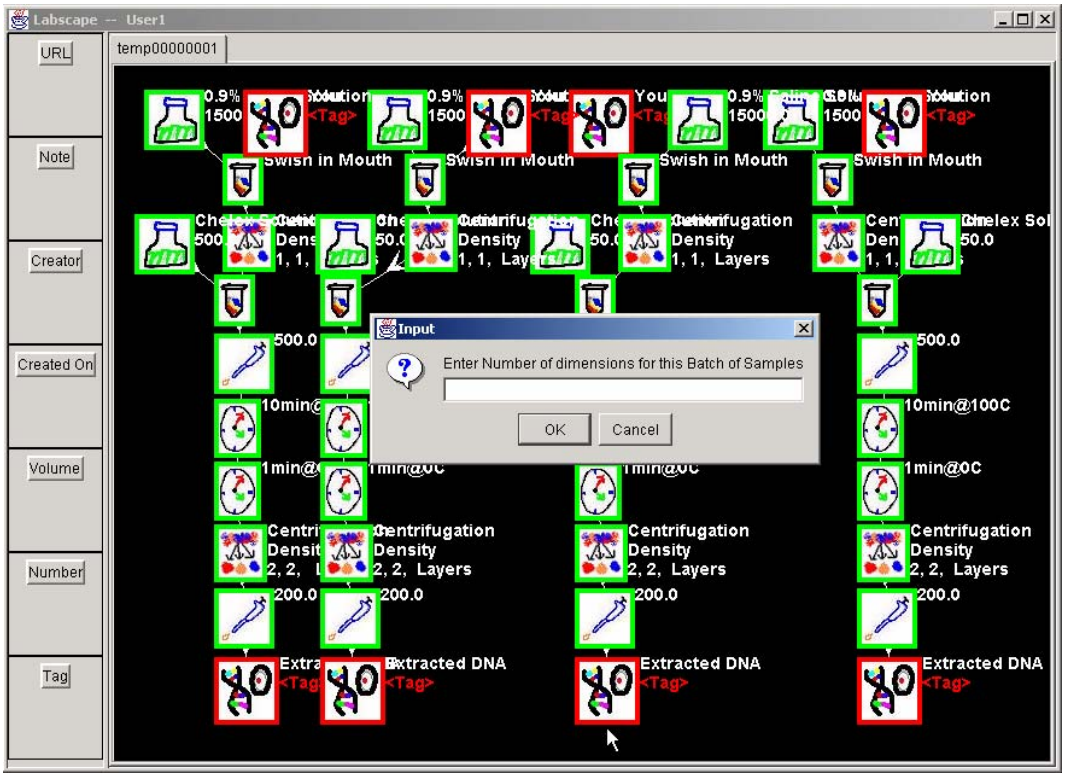

Figure 70: Text input dialog

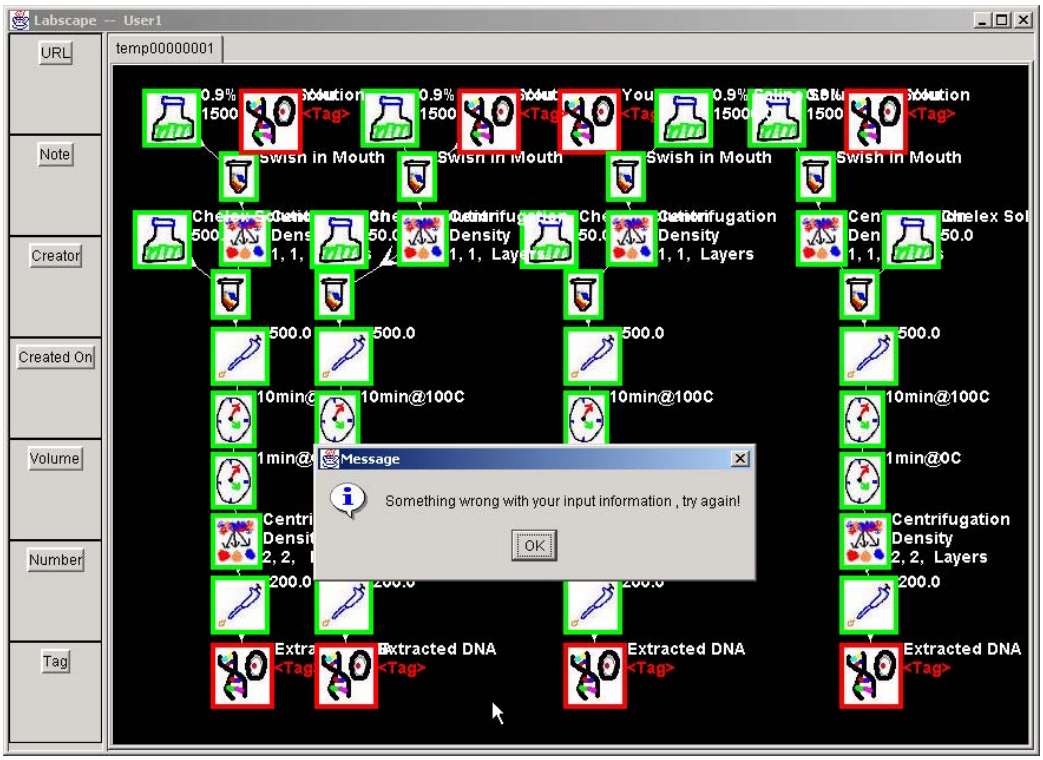

Figure 71: Error message in dialog window

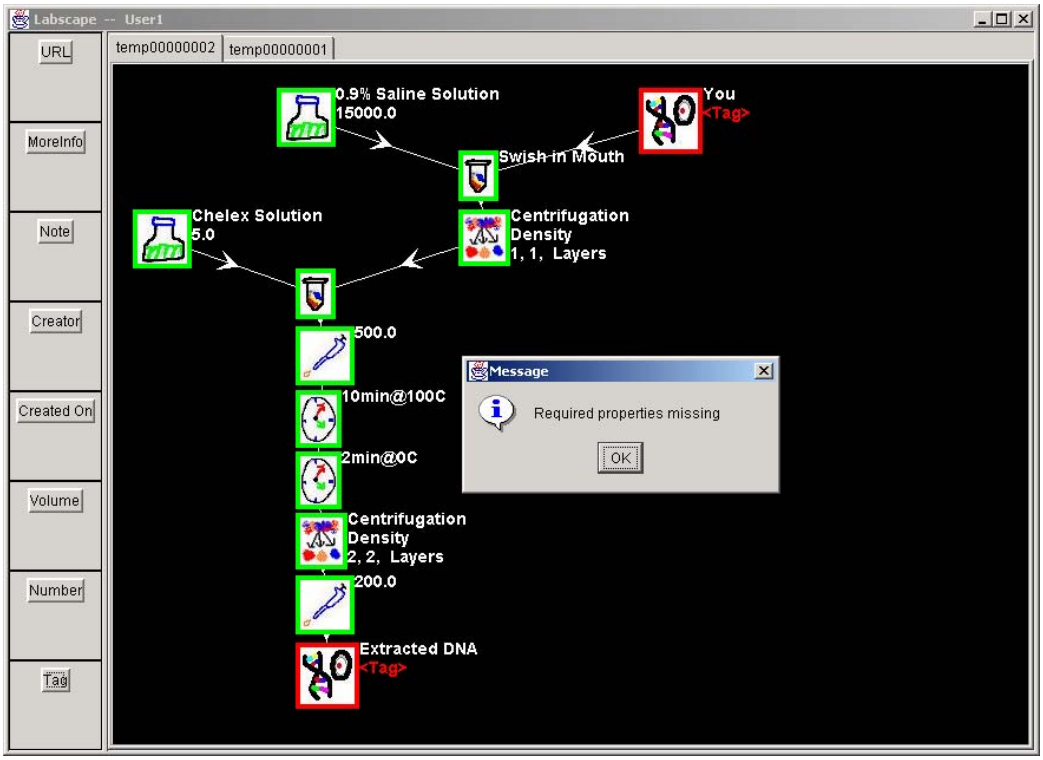

Figure 72: Error message in dialog window

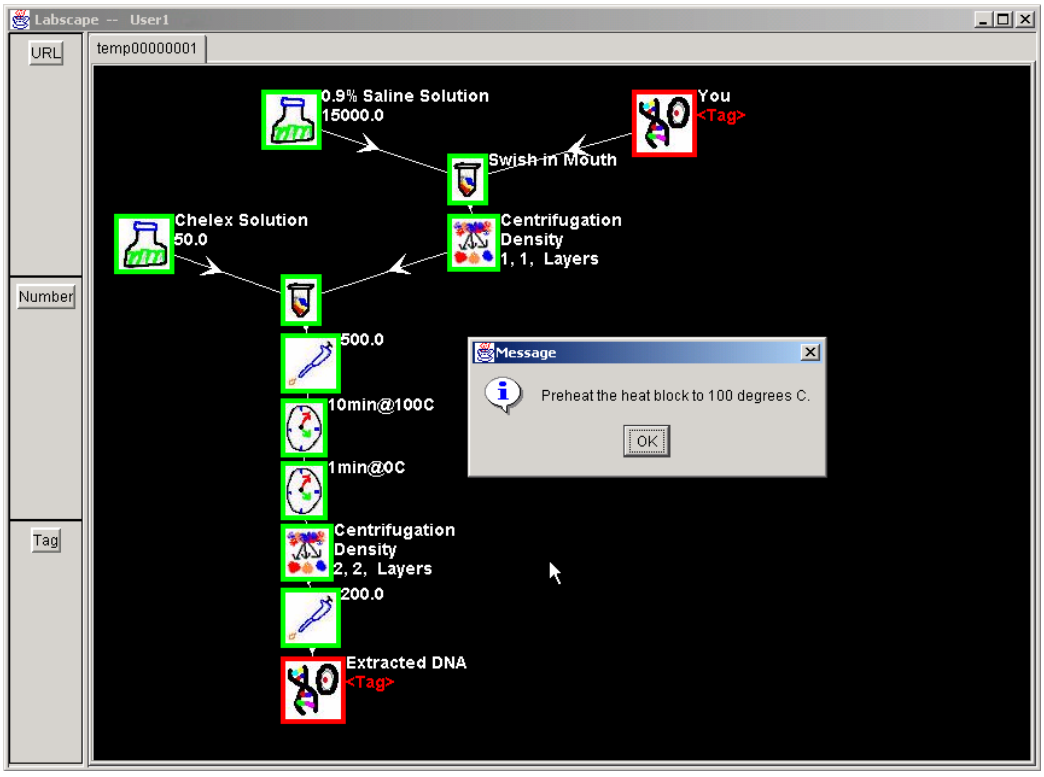

Figure 73: Informational message

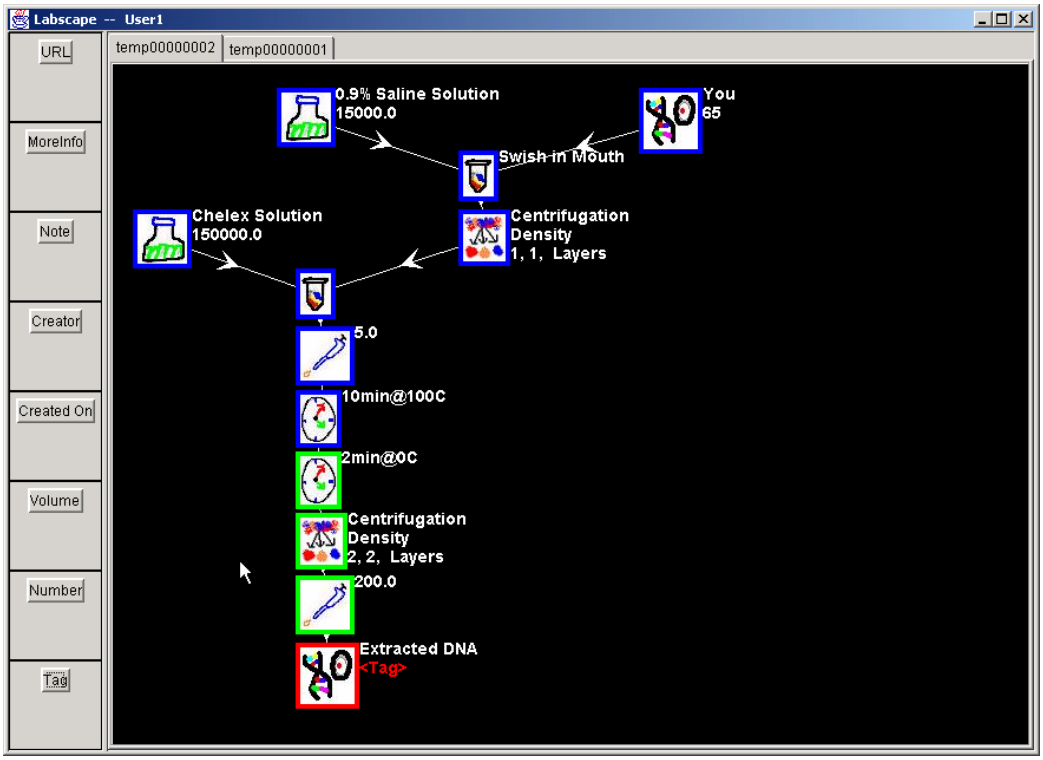

Figure 74: Left sidebar with many debug boxes

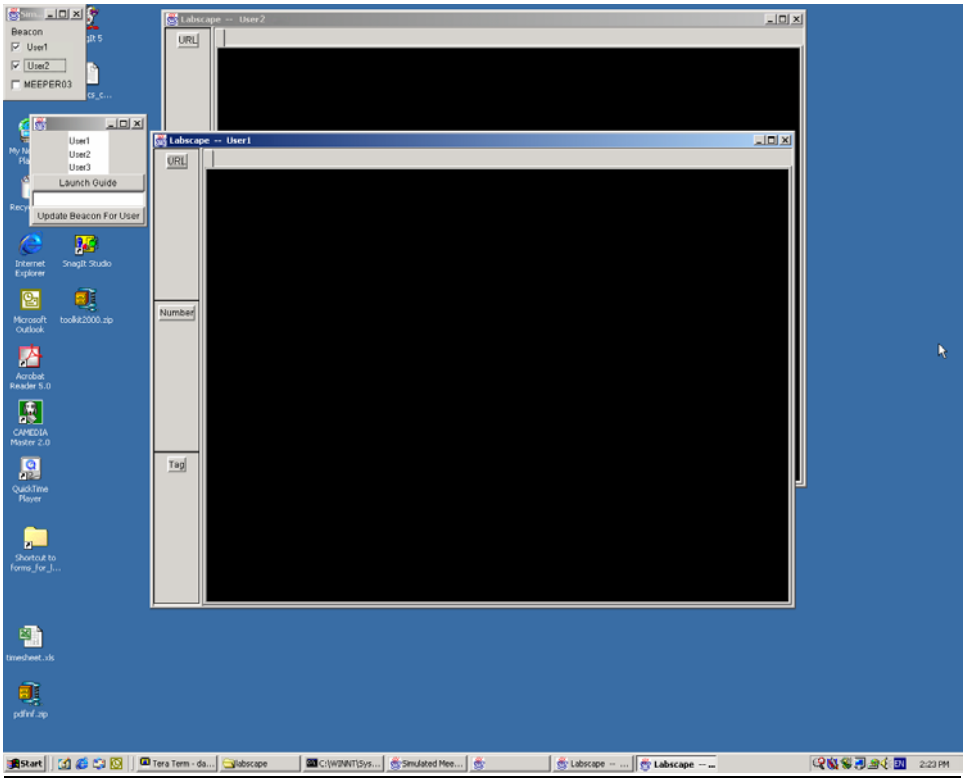

Figure 75: Two users logged in simultaneously

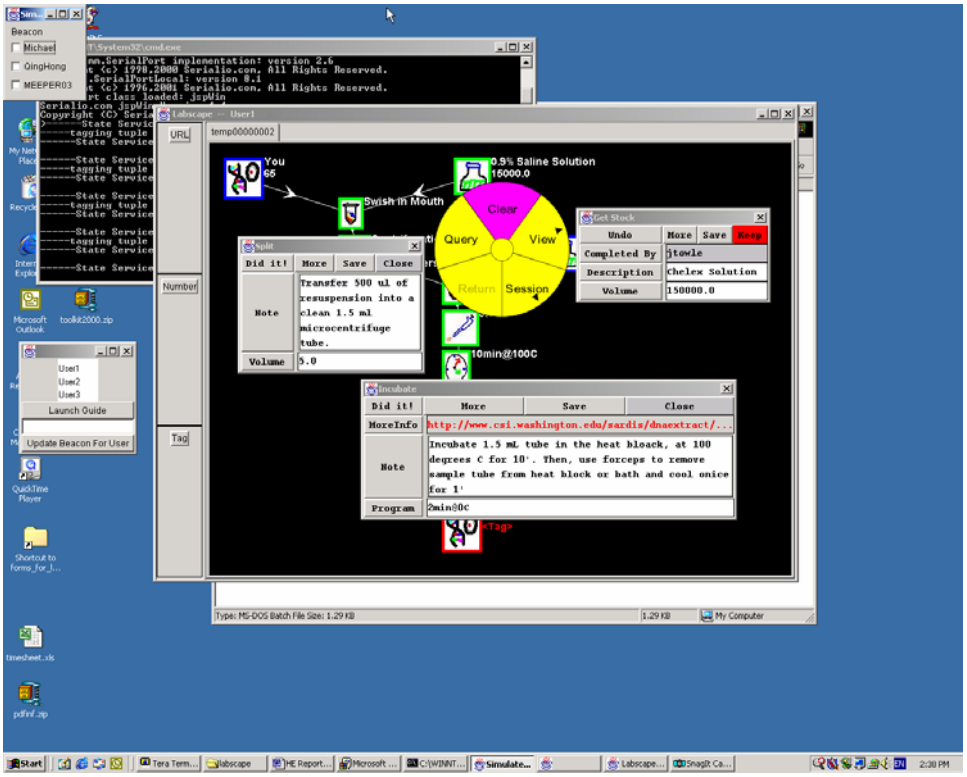

Figure 76: No user logged in, main window open (user unchecked name)

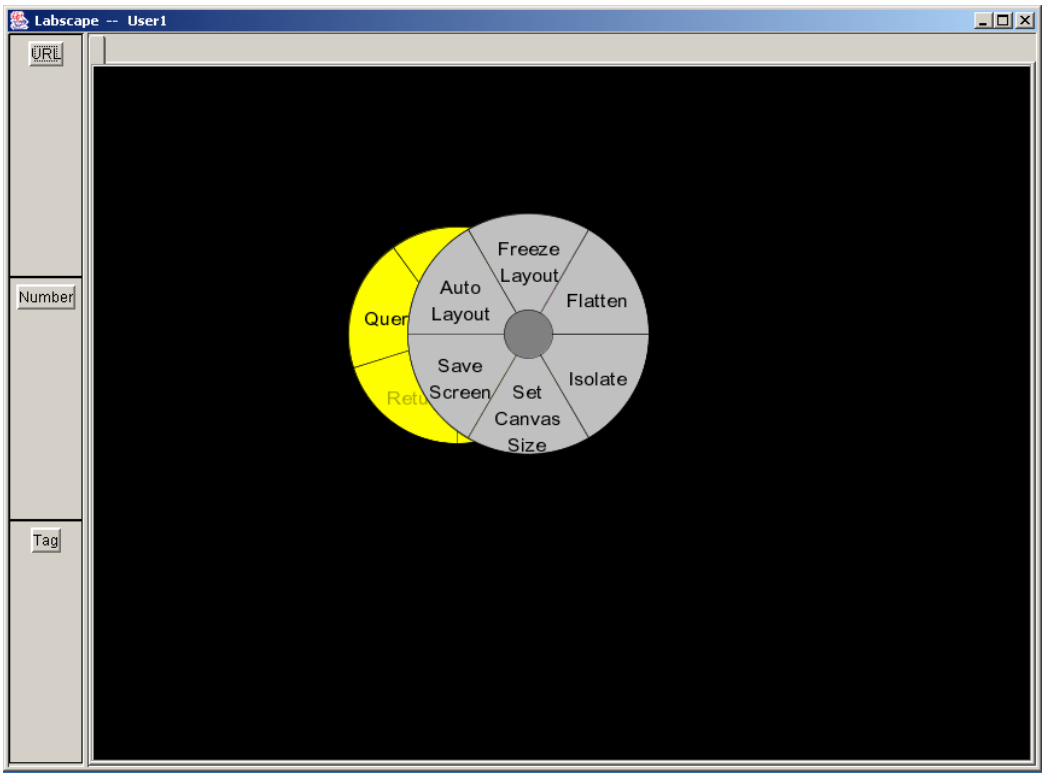

Figure 77: Gray-shaded *view* sub-menu

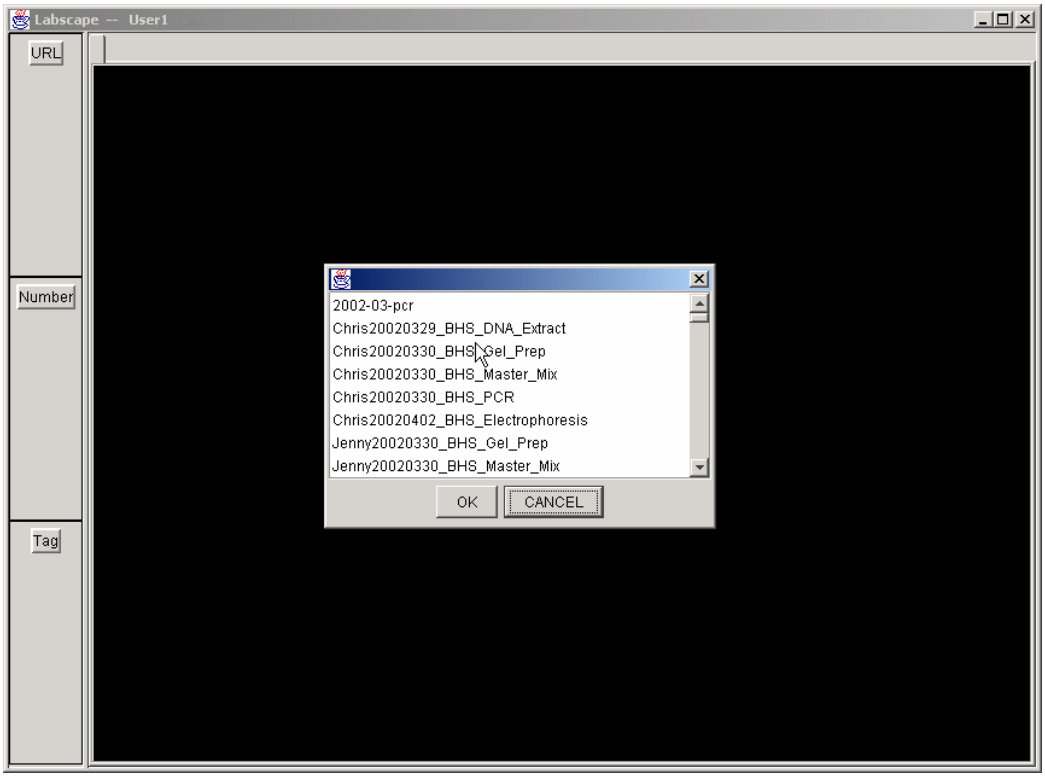

Figure 78: *Session selection window* (opens existing session)

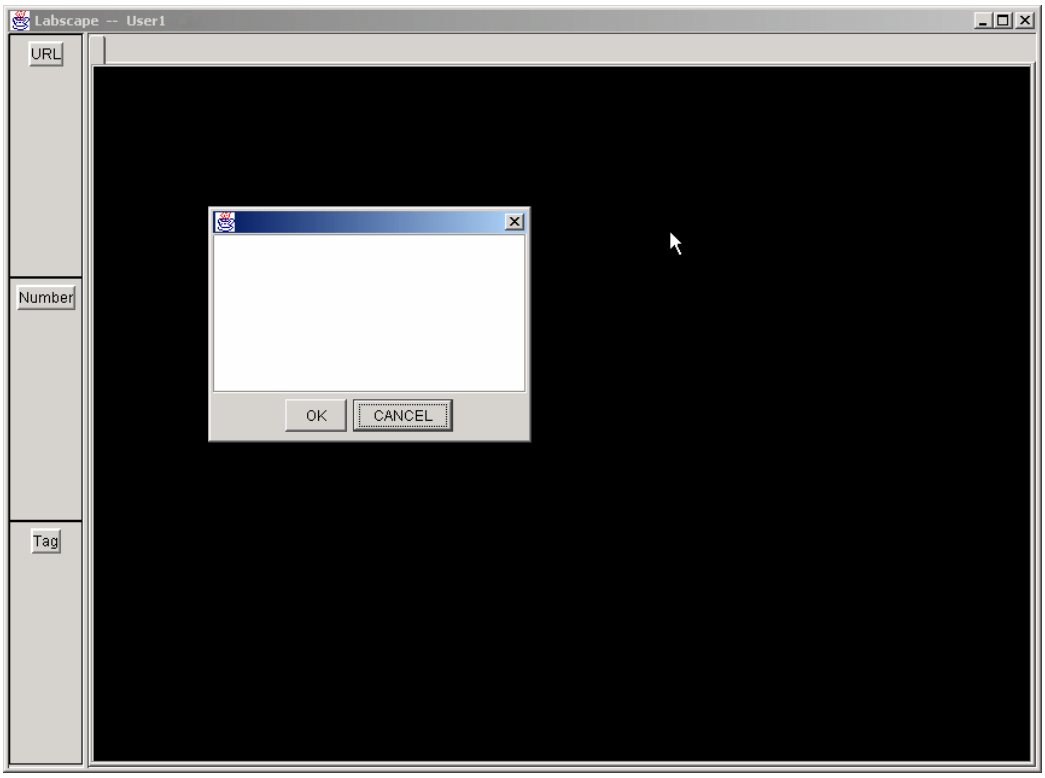

Figure 79: Empty *session selection window*

Appendix G: References

Nielsen, J. and Mack, R.L. (1994). *Usability Inspection Methods*. New York: John Wiley & Sons, Inc. 25-62.

Web-creators. (March 23, 2002). Heuristic Evaluation. Retrieved June 13, 2002 from: *http://www.stanford.edu/group/web-creators/heuristics.htm*